Business process discovery from emails, a first step towards business process management in less structured information systems

Marwa Elleuch

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAS014

Thèse de doctorat

TELECOM SudParis

IP PARIS

Business Process Discovery From Emails, A First Step Towards Business Process Management In Less Structured Information Systems

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Evry, le 20/12/2021, par

Marwa ELLEUCH

Composition du Jury :

M. Mohamed QUAFAFOU Professeur, Université de Aix-Marseille, France	Président
M. Hajo A.REIJERS Professeur, Université de Utrecht, Pays Bas	Rapporteur
Mme. Daniela GRIGORI Professeur, Université Paris-Dauphine, France	Rapporteure
M. Abderrahmane MAARADJI Maître de conférences, ECE Paris Lyon, France	Examinateur
M. Vincent CLAVEAU Professeur, IRISA, France	Examinateur
M. Walid GAALOUL Professeur, Télécom SudParis, France	Directeur de thèse
Mme. Oumaima ALAOUI ISMAILI Docteure, Orange Labs, France	Co-encadrante de thèse
M. Nassim LAGA Docteur, Orange Labs, France	Co-encadrant de thèse

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAS014

Thèse de doctorat

Business Process Discovery From Emails, A First Step Towards Business Process Management In Less Structured Information Systems

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis – Orange Labs

École doctorale n°626 Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Evry, le 20/12/2021, par

Marwa ELLEUCH

Composition du Jury :

M. Mohamed QUAFAFOU Professeur, Université de Aix-Marseille, France	Président
M. Hajo A.REIJERS Professeur, Université de Utrecht, Pays Bas	Rapporteur
Mme. Daniela GRIGORI Professeur, Université Paris-Dauphine, France	Rapporteure
M. Abderrahmane MAARADJI Maître de conférences, ECE Paris Lyon, France	Examinateur
M. Vincent CLAVEAU Professeur, IRISA, France	Examinateur
M. Walid GAALOUL Professeur, Télécom SudParis, France	Directeur de thèse
Mme. Oumaima ALAOUI ISMAILI Docteure, Orange Labs, France	Co-encadrante de thèse
M. Nassim LAGA Docteur, Orange Labs, France	Co-encadrant de thèse

Acknowledgments

I would like to acknowledge all people that have supported me during my work on my thesis.

I would like to thank all members of the jury. I thank Professor Hajo A.REIJERS and Professor Daniela GRIGORI for accepting being my thesis reviewers and for their attention and thoughtful comments. I also thank Professor Mohamed QUAFAFOU, Professor Vincent CLAVEAU and Dr. Abderrahmane MAARADJI for accepting being my thesis examiners.

I would like to express my appreciation and gratitude to my supervisors: Dr. Oumaima ALAOUI ISMAILI, Dr. Nassim LAGA and Professor Walid GAALOUL for their frequent encouragement, their valuable advice, constant support and for their relevant guidance and feedback that have helped me during my thesis. I thank them for their constant involvement in my work, for their eye for details, also for being willing for listening me, coaching me and sharing their knowledge with me which were always stimulating and greatly contributed to progress in my thesis.

I would like to thank Oumaima ALAOUI ISMAILI for being not only an advisor but also a good listener and a friend who has been by my side in most of the moments. I would like to thank Nassim LAGA whose leadership, research and managerial expertise allowed me to acquire new understandings in the process mining field and better manage my PhD project. I equally thank Oumaima ALAOUI ISMAILI and Nassim LAGA for their motivation and for bringing their algorithmic and logical way of thinking in my work. I would like to thank Walid GAALOUL for bringing his scientific and managerial expertise in directing my thesis. I thank him for his continuous interest in my work, his long term vision and his mastery of his field of research, which have been the key to value my work into scientific contributions accepted in renowned conferences in the process mining field.

I am also thankful to Dr. Nour ASSY and Professor Boualem BENATALLAH for the beneficial collaboration we have had. A special thanks go to the anonymous reviewers for their constructive feadback.

I continue with expressing my gratitude to Orange Labs and its management for funding my doctoral studies. I am grateful for the members of the SIDE team that I integrated within Orange Labs for their constant support. I namely thank Jérome DESCOS, the manager of the SIDE team, and Philippe LEGAY, the responsible of the Orange Process Discovery Program, for having faith in me.

Finally yet importantly, I would like to thank my family for their countless blessings and unwavering support all the time. I would also like to thank all my friends, namely Wided HAMMEDI and my neighbors Marwa KAZDOGHLI LAGHA, Chaiema EL FAKIR, Imen BOUABIDI and Yousra HSINA who were always there for me and also for the lovely moments we spent and for the deep discussions we had. I equally thank them for having freed themselves and made the trip to Paris just to attend my thesis defense and to share with me

the last moments of my thesis journey.

Contents

List of Figures

List of Tables

Table of notations

$\mathcal{ID}$	*The set of email identifiers*
$\mathcal{AD}$	*The set of email addresses*
email	*Email*
em	*Preprocessed email*
EmL	*Email Log Data*
Act	*Activity*
$\mathcal{A}$	*The set of activities*
$\mathcal{AN}$	*The set of activity names*
$\mathcal{BD}$	*The set of business data*
$\mathcal{BD}$	*The set of business context information*
$\mathcal{BI}$	*The set of business information*
$\mathcal{W}$	*The set of lemmatized words*
$\mathcal{T}$	*The set of part of speech category tags*
C	*Concept*
$\mathcal{C}$	*The set of Concepts*
$\mathcal{PC}$	*The set of patterns of concepts*
PC	*Pattern of Concepts*
EL	*Event Log*
ev	*Event*
SA	*Speech Act*
$\mathcal{SA}$	*The set of speech acts*
$\mathcal{ASA}$	*The set of activities specified by their speech acts*
$\mathbf{Act_o}$	*Activity Occurrence*
$\mathbf{AN_{occ}}$	*Activity Name Occurrence*
$\mathbf{BD_{occ}}$	*Business Data Occurrence*

$\mathbf{BC_{occ}}$	*Business Context Occurrence*
$\mathcal{RI}$	*The set of relevant information values*
Th	*Thread*
$\mathcal{TH}$	*The set of threads*
At	*Actor*
$\mathcal{AT}$	*The set of actors*
Ar	*Artifact*
$\mathcal{AR}$	*The set of artifacts*
ζ	*The set of distributions*
$\mathcal{G}_{\mathbf{SR}}$	*The set of Sender-Receivers groups of activities*
$\mathcal{G}_{\mathbf{At}}$	*The set of actor groups*
Const	*A sequencing constraint*
$\mathcal{CT}$	*The set of constraint types*
$\mathcal{AS}$	*The set of associations*
$\mathcal{ARC}$	*The set of artifact cardinalities*
MC	*Multi-meaning Concept*
MI	*Multi-meaning Intersection*

Chapter 1

Introduction

Contents

1.1 Context and Motivation

The recent spread of digital transformation within organizations has led to the replacement of manual tasks by digital ones. This includes the integration of information systems (IS) at different organizational levels (i.e. operational, knowledge, management and strategic) to automate more tasks in order to gain in productivity and efficiency. IS are combinations of hardware, software, and telecommunications networks that people build and use to collect, create, and distribute useful data [89]. At operational level, IS are used to support the processing of the daily business transactions/operations. At the knowledge level, IS are used to support office and managerial activities (e.g. printing, mailing, scheduling meeting) or to make organization business knowledge available for its workers (e.g. through research engines, friendly visualization tools). At management level, IS support the planning, controlling, and decision-making functions. Finally, at the strategic level, they assist senior managers to monitor performance, track activities of competitors, identify opportunities and forecast trends.

According of Silver et al. [85], IS are made of five components:

- Hardware: It consists of input/output device, processor, operating system and media devices;

- Software: It consists of various programs implementing a set of instructors telling the hardware what to do;
- Data: They refer to the collection of facts (e.g. address, phone number, client names) that are used by IS to produce useful knowledge. They are stored in human readable format (e.g. database);
- People: They refer to users, and persons who operate and service the computers, those who maintain the data, and those who support the network of computers;
- Processes: They refer to the policies that govern the operation of an IS;

The last point highlights the importance of processes in IS. A process is a series of steps undertaken to achieve a desired outcome or goal. IS are becoming more and more integrated with organizational business processes (BP) that are focused on achieving business goals. BP are of big importance in organizations. They describe and document the operational mode of organizations allowing better transparency, management and optimization of the performed operations and the required resources. Additionally, they are among the main requirements for obtaining quality management certifications necessary to be internationally recognized and to gain client confidence. The ultimate goal of IS is not simply automating BP steps at operational level, but also to manage and improve them. The field of Business Process Management (BPM) [26] have appeared to ensure continuous BP improvement.

BP mining [90] is a family of techniques that were introduced to support the analysis of BP from IS traces in order to improve BPM. The goal of BP mining is to turn IS traces into valuable insights on BP through three different types of analysis:

- Discovery: It aims to identify models reflecting how BP are actually performed within IS;
- Conformance checking: It compares actual execution of BP with their reference model. The goal is to check if the actual execution is compliant with the theoretical one and to detect if there are deviations or bottlenecks;
- Extension: It aims to enrich reference models of BP on the basis of their actual executions. The goal is to perform further updates and modifications on BP implementations to be more compliant with the reality;

BP discovery is an elementary task that ensures the different types of analysis. For instance, it is required to identify the actual BP execution. This allows to check their conformity with a reference model and to identify extensions/improvements to be integrated in the actual implementations. Existing BP discovery approaches [78, 14, 99, 99, 27, 67] are usually limited to analysing execution logs of IS intended for supporting the execution of BP activities (in distributed or in centralized way). Such execution logs are supposed to be of high or of middle level of maturity w.r.t BP [91], which means that: (i) they are composed of structured records while each one captures evidence of one BP activity execution and, (ii) a part of

events' attributes (e.g. activity name, timestamp) are explicitly included in the execution records which facilitates their inference. This allows them to be pre-processed into structured event logs compatible with existing BP discovery algorithms [96, 37, 3, 1]. Each one of these event logs is supposed to contain events reporting all the execution traces of a single and a predefined BP through the following elements [90]:

(i) Activities referring to the performed elementary BP step at each event;

(ii) Timestamps indicating when an event was occurred;

(iii) Instance identifiers (ID) relating multiple events to the traces of specific BP executions. Each trace is supposed to cover a complete BP execution. Given the example of a recruitment BP, all events carried out in the context of a job instance J_1 (e.g. data scientist) form one trace T_1. Equally, all events carried out in the context of another job instance J_2 (e.g. development engineer) form another trace T_2 ;

(iv) Actors referring to who have performed the events' activities;

(v) and optionally, Informational entities defining the data consumed and produced with respect to each event.

Such event log structure has been used to discover three BP perspectives types:

(i) Actors (i.e. organizational) perspective to describe who (e.g. people, departments) is involved in BP activities [93, 87, 71, 81];

(ii) Data perspectives to identify the informational entities manipulated by BP activities [58, 56];

(iii) Behavioral perspective to specify the conditions for activities execution [5, 21];

The traces of other IS that are not intended for automating the operational mode of BP but that could support the execution of BP/BP parts informally were generally ignored. Particularly, emailing systems are widely involved as an alternative or as a complement to manage part of the business activities. They allow workers to share their expertise and to collaboratively perform their daily activities. The logs data of emailing systems could then report additional traces of BP executions, making them a valuable data source for BP discovery and for automating the collecting of the undocumented BP expertise. Let take the example of the recruitment process, all contacts with candidates go through emails for sending/forwarding resumes, planning interviews or informing about hiring decisions. In other contexts, emails could be used for handling some specific events while performing BP. Figure 1.1 includes an example of real emails retrieved from Enron database[1]. It shows a set of interactions between employees outside a gas trading system for handling a flow gas event (i.e. a gas volume that exceeds a certain threshold). The emails are related to the same

[1] http://www.cs.cmu.edu/~enron

gas meter (Meter 5192) and to the same trading deal (deal 454057). They belong to two conversations (of subjects 'Flow w/no nom' and 'Dec 00') and are sorted in ascending order according to their timestamps. The emails report how employees act when a meter detects a gas flowing. The first and the third emails show how an employee notifies his manager once this event occurs to request the execution of some activities (e.g. extend the associated deal or create a new one in email $email_3$). The second and the fourth emails report the activity carried out by the manager to cover the flowing event of the gas meter (roll or extend the associated deal as indicated in $email_2$ and $email_4$ respectively). Figure 1.1 illustrates then a case where a trading BP part related to managing gas deals is supported by emails inside Enron company. It shows also an example of an email (i.e. $email_3$) summarizing employee expertise when handling some events/exceptions (through the requested activities).

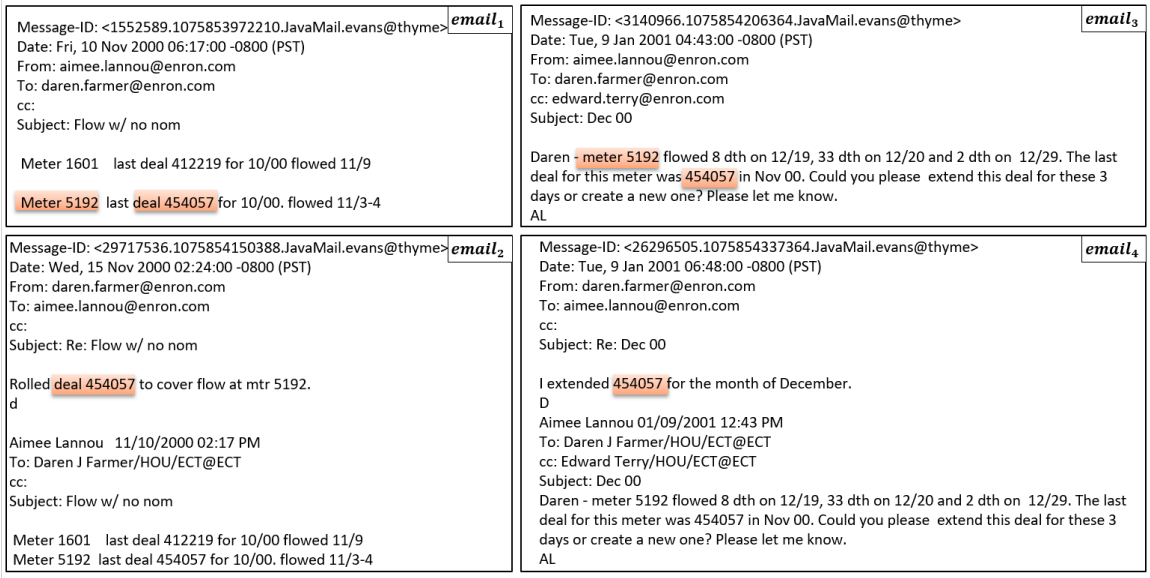


Figure 1.1: Emails retrieved from Enron data-set

From a BPM perspective, performing activities through emails facilitates interactions between the different BP actors. This could speed up the process of making decisions or handling exceptions or problems by customizing the execution of some BP parts. However, this equally induces some inconveniences, especially in the absence of tools allowing the discovery of the related execution traces. On the one hand, these execution traces are likely to be unmanaged due to the non exploitable format of their storage. They are included in the overall email log data and their related relevant information (e.g. activity name, BP name, manipulated business data, etc) are not explicitly separated/identified to be stored according to a specific structure. Therefore, they could not be easily reused when it is needed or exploited to track BP executions. On the other hand, manual intervention remains required when performing BP activities through emails. If performed repeatedly, they are likely to be time consuming, which leads to the need to automate their execution.

BP discovery from emails is then important to improve BPM. However, given the non-controlled nature of emailing systems, traditional BP discovery algorithms could not be ap-

plied (or at least not directly applied) on emails for three main reasons. First, emails do not have the suitable structure as they are of unstructured textual nature; each email is not explicitly mapped to its related BP elements as the case of an event in a classic event log. Email log could also concern more than one BP that are not known a priori (i.e. with undefined functional perspective). Second, an email containing an activity does not necessary mention all the related BP elements. This induces data insufficiency for inferring the attributes of each event (e.g. the exact timestamp of the performed activity, the related instance ID, etc). Even in the case of mentioning the related BP elements, they would not be written in a regular way by emails' users (i.e. the same BP element could be expressed differently). Third, BP that could be discovered from emails differs from those that could be discovered from classic event logs due to the following reasons:

- ***The richer content of emails:*** Emails could be used by employees for various purposes and not only for executing activities as the case of BP management systems, e.g. requesting the activity execution, informing about the activity execution, etc. Taking into account these various types of use, BP events could be more specified to better reflect all BP insights in emails. Additionally, employees contributions (e.g. request, planning, execution) could be inferred to enrich BP actor perspective.
- ***The less structured BP introduced in emails:*** Employees actually introduce in emails **poorly structured BP fragments** (i.e. parts) with high variability rather than well-structured and complete ones. For instance, given that emailing systems are of non-controlled nature, this implies that there are no (or less) rules to be respected by employees when writing BP elements in emails. Additionally, there are no restrictions that could limit or requires a minimum amount of knowledge to be introduced at each BP instance. This would lead to obtain, for one BP, a set of incomplete traces whose activities could vary form one instance to another. Therefore, only the trace of BP fragments (i.e. BP parts) could be found in common between different communication traces of the same BP.

The context of this thesis revolves then around the analysis of emails for discovering BP fragments. This mainly allows BP elements present in emails to be better managed by organization workers. By mapping emails to BP events, additional traces of existing or undocumented BP would be reported to provide a more complete view on their executions. This would enhance BPM by allowing:

- More complete monitoring of the execution of BP activities; execution problems and bottlenecks could be identified from employees' communication traces in emails;
- The discovery of the undocumented BP or BP parts whose execution relies on workers' implicit knowledge/expertise; this enables their automation to gain more efficiency;
- The usability of previous BP execution traces to assist workers when performing future activities through emails concerning the same or different BP instances. Workers would have better visibility on what have happened in past BP/activity executions. They

could access in an easier way to the related information through any system managing the past traces (e.g. a recommendation system that brings out from these traces relevant BP knowledge related to activities being performed through emails).

Due to the unstructured nature of email log data, discovering BP from emails arises several research problems comparing to the case of discovering them from structured event logs. In what follows, we discuss some of these research problems in Section 1.2. We outline then our thesis objectives, principles and contributions in Section 1.3. Finally, we present the structure of the thesis in Section 1.4.

1.2 Research problem: How to discover BP from the unstructured log data of emails ?

We faced two important challenges towards the discovery of BP from emails. The first challenge is related to the nature of the knowledge we aim to discover. Indeed, as we detailed earlier, the nature of emails (richer and less structured) induces several differences of BP executions compared to their execution in a traditional IS. The first research question is thus **(Q1) What are the BP knowledge (i.e. BP perspectives and BP elements) that could be discovered from emails and how to formalize them ?**
The second challenge is related and induced by the nature of our input. Indeed, emails are of non-structured nature (w.r.t BP). The subject and the body of emails, which contain significant BP knowledge, are of free text. The second research question is thus **(Q2) How to discover BP knowledge from emails ?**

These research questions can be further divided into several sub-questions. Some of them were previously discussed in related studies and others were not previously handled, but could be deduced from the unstructured nature of email logs data when compared to the structured and the precise nature of BP. We separate these research sub-questions according to Q1 (Section 1.2.1) and Q2 (Section 1.2.2).

1.2.1 What are the BP knowledge that could be discovered from emails and how to formalize them ?

We mean by BP knowledge, the different BP perspectives (e.g. data, actor) or BP elements (e.g. activity, event) to be discovered after analysing emails. Existing studies that have discovered BP knowledge from emails were generally limited to the discovery of BP functional or/and behavioral perspectives [46, 24, 22, 55, 11, 52]. Additionally, they have targeted the discovery of BP elements present in the most common event logs, or sometimes, they were limited to the discovery of business activities [25, 50, 49, 65, 54, 18, 76, 77, 43, 86, 48].

Nevertheless, BP knowledge that could be discovered from emails differ from that which could be discovered from the execution logs of IS supporting BP activities. For instance,

emailing systems are not used by workers in the same way as these IS when performing BP; an IS supporting BP activities could direct workers through forms/checklists to bring some kind of regularity to the typed information. On the contrary, an emailing system allows workers to type their messages freely. In other words, IS supporting BP activities could be viewed as controlled IS that follow a set of implemented policies and directives. These policies could be in relation to a predefined BP in the case of BPM systems (BPMS) or more generally, in relation to how an activity should be performed. This has the impact of restricting and structuring the information typed by workers. As for emailing systems, they could be viewed as non-controlled IS from business perspective. This means that they follow only the process that describe their operational mode (e.g. sending, receiving emails), but do not follow a priori known BP or activity to structure or restrict the typed textual contents of emails.

As consequence, email log data would be of less or of richer information compared to the execution logs of IS supporting BP activities. This is equally induced by the operating specificities of emailing systems which are different from those of such IS. Among these latters, there are :

- The conversational mode when exchanging emails between a group of interlocutors; One email could be sent as a response to a received email (e.g. $email_3$ and $email_4$ in Figure 1.1 form one conversation; $email_4$ is sent as a response to $email_3$);
- The presence of interlocutors lists in emails;
- The expression of BP elements in emails using natural language, named entities/numeric types or values;
- The possibility of using emails for different purposes in the context of BP activities that are not limited to execution purpose (e.g. as the case of BPMS). In fact, workers could use emails for informing about, planning or requesting an activity execution. Therefore, timestamps and senders of emails do not necessarily correspond respectively to the actual timestamps and performers of the activities expressed in these emails. Additionally, the events to be associated to the occurrence (i.e. appearance) of these activities in emails would differ from those referring to their execution. Taking the example of the emails $email_3$ in Figure 1.1, the sender informs about the event of flowing the meter gas. He also requests the execution of 'extend deal' or 'create deal' activities. This means that he is not the performer of these activities. Moreover, the timestamp of their related events do not correspond to the action of sending $email_2$: in the case of informed (requested) activities, timestamps of their events logically precede (follow) the email timestamp. In the case of some requested activities, they could be not performed in the future as the example of 'create deal' activity in $email_2$ (because the second employee preferred, in $email_3$, the extension of the deal from the requested activities).

In view of these limitations and emailing systems specifities, our first objective is to define BP knowledge that could be discovered from emails. To this end, we have to answer the following sub-questions:

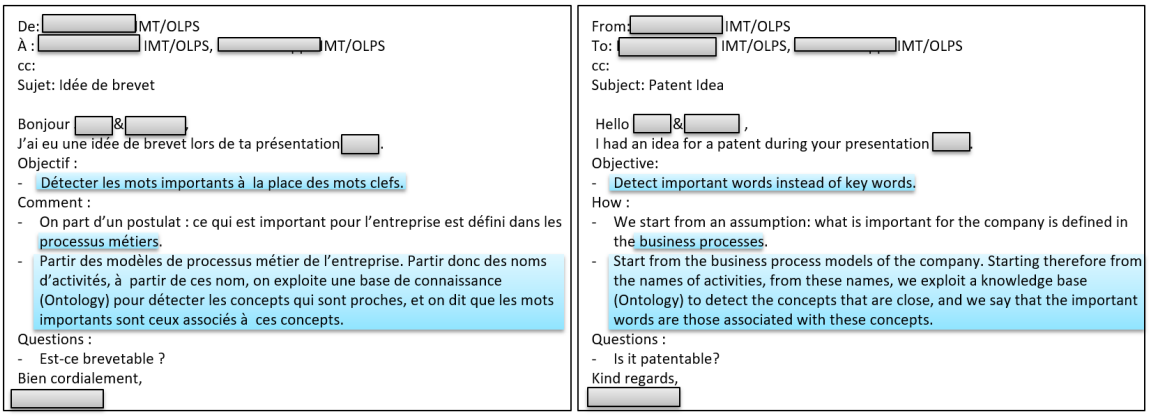
- Q1-1: How to define multiple BP perspectives while considering the specifities of using emailing systems ?
- Q1-2: What is the event log structure enabling the discovery of the defined BP perspectives ? i.e. What are the additional/less BP elements (compared to those of the most common event logs) to be retrieved from emails ?
- Q1-3: How to define the actual BP events related to activity occurrences (i.e. appearance) in emails taking into account the specifities of emailing systems ?

1.2.2 How to discover BP knowledge from emails?

Given the unstructured nature of email log data, BP elements could not be directly retrieved to allow the discovery of the defined BP perspectives. In view of this issue, it is necessary to start by discovering BP elements from emails. The goal is then to use the discovered BP elements to transform the unstructured email log data into a structured event log. This structured event log is to be mined to discover the different perspectives of BP. Nevertheless, this is far from trivial due to: (i) the informal way in which workers use emails, and (ii) the non-controlled content of emails. This means that there is no restriction on the expressed topics (business or non-business oriented), their number, their order of appearance or their granularity expression (e.g. number of words). Consequently, several issues could be highlighted:

1. **No a priori knowledge of email contents:** As workers freely type the textual content of their emails, they have more probability to express new and various business knowledge or to customize their BP activities. Therefore, we could not dispose a priori knowledge concerning the expressed BP elements in emails or the BP that emails were used in their context.
2. **Mixed content of emails:** From BP perspective, emails are of mixed content as they could contain relevant information related to BP elements or non-relevant information related to personal issues, greetings, etc.
3. **Various granularity expression levels of BP elements:** This expression granularity would concern two levels:
   - Precision level which means the precision of BP elements expressions, for example, the activity 'scheduling meeting' can refer to different kinds of meeting, related to different BP (e.g. project status meeting, candidate interview).
   - Quantity level which means the number of words used for expressing BP elements (e.g; a couple of words, one sentence, the overall email). This granularity would vary from one BP element to another and from one email to another. Taking the case of BP instance element, it could be expressed in emails through one word of numeric type (e.g. deal identifier in the email $email_1$ of Figure 1.1) or a set of words occupying a good part of the email, for example, in the context of activities intended for managing research projects, projects (which present here the instance

notion) could be introduced by a research engineer/responsible not only through their names, but also through their related topics. Figure 1.2 shows a real email sent in the context of proposing a patent for a BP research project within Orange labs. In the majority of the email (see textual content colored in blue), the employee uses project oriented vocabulary .
In other cases, the dominant topics could be related to BP as the example of the email $email_5$ of Figure 1.3 where a good part of email vocabulary belongs to the BP of trading power electricity.


De: [redacted] IMT/OLPS
À : [redacted] IMT/OLPS, [redacted] IMT/OLPS
cc:
Sujet: Idée de brevet

Bonjour [redacted] & [redacted],
J'ai eu une idée de brevet lors de ta présentation [redacted].
Objectif :
- Détecter les mots importants à la place des mots clefs.
Comment :
- On part d'un postulat : ce qui est important pour l'entreprise est défini dans les processus métiers.
- Partir des modèles de processus métier de l'entreprise. Partir donc des noms d'activités, à partir de ces nom, on exploite une base de connaissance (Ontology) pour détecter les concepts qui sont proches, et on dit que les mots importants sont ceux associés à ces concepts.
Questions :
- Est-ce brevetable ?
Bien cordialement,
[redacted]

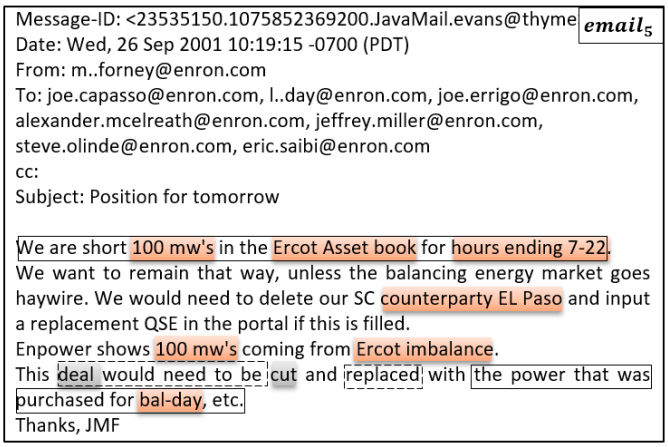
(a) Original email (written in French)

From: [redacted] IMT/OLPS
To: [redacted] IMT/OLPS, [redacted] IMT/OLPS
cc:
Subject: Patent Idea

Hello [redacted] & [redacted] ,
I had an idea for a patent during your presentation [redacted].
Objective:
- Detect important words instead of key words.
How :
- We start from an assumption: what is important for the company is defined in the business processes.
- Start from the business process models of the company. Starting therefore from the names of activities, from these names, we exploit a knowledge base (Ontology) to detect the concepts that are close, and we say that the important words are those associated with these concepts.
Questions :
- Is it patentable?
Kind regards,
[redacted]

(b) Email Translated to English with www.DeepL.com/Translator

Figure 1.2: Real email for proposing a patent in the context of a research project within Orange labs


Message-ID: <23535150.1075852369200.JavaMail.evans@thyme> $email_5$
Date: Wed, 26 Sep 2001 10:19:15 -0700 (PDT)
From: m..forney@enron.com
To: joe.capasso@enron.com, l..day@enron.com, joe.errigo@enron.com, alexander.mcelreath@enron.com, jeffrey.miller@enron.com, steve.olinde@enron.com, eric.saibi@enron.com
cc:
Subject: Position for tomorrow

We are short 100 mw's in the Ercot Asset book for hours ending 7-22.
We want to remain that way, unless the balancing energy market goes haywire. We would need to delete our SC counterparty EL Paso and input a replacement QSE in the portal if this is filled.
Enpower shows 100 mw's coming from Ercot imbalance.
This deal would need to be cut and replaced with the power that was purchased for bal-day, etc.
Thanks, JMF

Figure 1.3: Real email retreived from Enron dataset for planning trading positions

It is important to note that even in the case of having two expressions with the same precision level, they could be of different granularity at quantity level. For instance, as each email user (e.g. employee) could have his own writing style when expressing his/her ideas, these expressions could be expressed differently through words sharing the same meaning or the same context of use. Moreover, they could differ due to the use of specific business abbreviations (adopted internally within an organization or by a set of workers), each one resumes a set of words into one word.

4. **The presence of multiple BP elements in one email:** As the expression of one BP element does not necessary fit the overall textual content of one email, multiple BP elements could appear in the same email of the same type (e.g. multiple activities) or of different types (e.g. activity, instance, informational entities). Taking the example of the email $email_5$ in Figure 1.3; the sender introduced at least four activities for: (i) opening a long trading position (framed by continuous line in the first sentence), (ii) cutting a deal (colored by grey), (iii) replacing a deal (framed by dashed rectangle), and (iv) purchasing energy (framed by continuous line in the last sentence).
5. **Uncertainty of BP elements presence in emails:** In the case of expressing an activity in an email, there is no guarantee that the values of all the related BP elements will be provided. Taking the example of the instance identifier, it was explicitly mentioned in the emails $email_1$, $email_2$, $email_3$ and $email_4$ (i.e. deal 454057) in Figure 1.1 while it was absent in the email $email_5$ of Figure 1.3. As for the timestamps and the actors of the expressed activities, they are rarely mentioned in emails in explicit way. We could find some indicators of times/actors in some cases (e.g.the personal pronoun 'I', 'you') as they could be totally ignored in others (e.g. activities expressed in the last sentence in Figure 1.3).

So far, some studies have been proposed to discover BP knowledge from emails. However several limitations could be highlighted when referring to the above issues.
First, they generally intended to convert email log data into the most common event log structure and then to apply existing process discovery algorithms compatible with their analysis [96, 37, 3, 1]. Actually, this is limited as these process discovery algorithms do not cover the additional/fewer BP elements present in emails.
Second, they mostly relied on integrating supervised learning techniques [18, 55, 76, 48] or manual selection/organization of the relevant information [86]. This needs knowing BP elements in advance which is not always feasible and involves human intervention (e.g. for labeling training datasets) [48]. Additionally, this limits the knowledge that could be discovered from emails to the predefined one.
Finally, they generally suppose that one email corresponds to the execution of one activity when generating event logs [4, 44, 46, 55]. Such assumption is actually limited as it does not consider BP element multiplicity per one email and the various purposes when using emails in the context of BP activities. It is important to note that activity multiplicity per one email was until now studied in the context of activity discovery approaches [77, 43, 48] and only one work [47] has considered it in the context of BP discovery.

In light of these issues and limitations, our second objective is to automate the discovery of BP knowledge from emails considering their non-controlled and non-structured nature. With the aim of addressing this research need, we have to simultaneously answer the following sub-questions:

- Q2-1: How to discover BP elements without disposing a priori knowledge about them ?
- Q2-2: How to select, from emails, relevant information in relation with BP elements ?

- Q2-3: How to discover BP elements considering the different degree of their granularity expression ?
- Q2-4: How to discover BP elements considering their multiplicity in emails ?
- Q2-5: How to deal with the uncertainty of BP element presence in emails (e.g. for estimating the relative order of events in the absence of their exact timestamps) ?
- Q2-6: How to mine the generated structured event log to discover multiple BP perspectives ?

1.3 Thesis objectives, principles and contributions

1.3.1 Thesis Objectives and principles

In the light of the previously described research problems, the main objectives of this thesis are summarized as follow:

- **Objective 1:** Define the BP knowledge (i.e. BP perspectives and BP elements) that could be discovered from emails;
- **Objective 2:** Automate the discovery of the defined BP knowledge from emails:
  - Without disposing a priori knowledge about them;
  - While allowing the discovery of multiple BP elements per one email;

  This objective requires the achievement of two sub-objectives:
  - **Objective 2.1:** Automate the generation of a structured event log from emails;
  - **Objective 2.2:** Automate the discovery of BP fragment perspectives from the generated event log;

To this end, we consider the following principles:

- **Principle 1: Consistency:** The BP knowledge to be discovered and the discovery approach must be consistent with the specifities of emailing systems regarding: (i) their operational mode, and (ii) their use in the context of BP;
- **Principle 2: Automation:** The discovery approach should propose automated techniques in order to discover the different BP knowledge from emails. These techniques should not require significant and time consuming human intervention;
- **Principle 3: Data robustness:** The discovery approach should propose techniques able to analyse data not limited to predefined use cases (e.g. emails related to a selected BP) or email types (e.g. emails written/generated according to textual template that are a priori known, for example, emails sent by robot systems);

- **Principle 4: Balanced Recall:** The discovery approach should make a compromise between the recall of the discovered BP knowledge and their degree of significance;

It is noteworthy that the proposed work in this thesis needs to be: (i) validated through public dataset of emails to allow comparison with related studies, and (ii) evaluated through different experiments on real datasets. Furthermore, the implementation, experiments, and results should be detailed.

1.3.2 Thesis contributions

To meet the above objectives while handling the described research issues, we introduce several algorithms to build a framework intended for discovering multiple BP perspectives from an email log data. The framework is composed of four main components enabling: (1) Pre-processing an email log data, (2) Converting it into a structured event logs compatible with multi-perspectives BP discovery, (3) Mining the structured event logs to discover BP w.r.t their functional, data, actor and behavioral perspectives, and (4) Recommending and visualizing BP oriented information to workers.

The contributions of our work can be summarized as follows:

- **Define what type of BP knowledge could be discovered from emails and formalize their definition:** We carried out an exploratory task to identify the specifities of using emails in the context of BP compared to using IS supporting their activities. These specifities were employed to define four BP perspectives: functional, actor, data and behavioral (**Q1**, **Principle 1**);
  For the functional perspective, we adopt the notion of 'BP fragment' (i.e. a set of BP elements defining a BP part) to be discovered from emails rather than complete BP due to the incompleteness of BP traces in emails.
  For the actor perspective, we take into account two specifities of emails (i.e. email interlocutors and their purposes when expressing BP activities in emails) to define the following BP elements:
  - Activity actors inferred from email interlocutors;
  - Activity groups of actors inferred from email interlocutors frequently interchanging activity related emails;
  - Actor contributions inferred from senders' purposes of expressing activities in emails;

  For the data perspective, we define the BP element artifact as the informational entities manipulated by activities while considering the named entities and numeric values present in emails textual content;
  As for the behavioral perspective, we integrate the notion of the purpose of expressing an activity in an email. We define three event types forming sequencing constraints of three behavioral models (**Q1-3**). These event types define from which perspective

we can view the occurrence of an activity in an email. They could correspond to: (i) the name of the occurred activity, (ii) The sender purpose of expressing an activity in an email, or (ii) the combination between activity name and the sender's purpose of expressing it in the email to better reflect how emails are used in the context of BP;

To formalize the relationship between an email log data and these different BP perspectives, we propose a meta-model composed of four parts: (i) Input part to describe the unstructured email log data, (ii) Output part to describe BP perspectives, and (iii) First intermediate part for describing the structured representation of emails in the form of an event log enabling multiple BP perspectives discovery, and (iv) Second intermediate part for describing entities appearing in the pre-processing phase towards obtaining a structured event log. We define by this way a research road-map for an effective multi-perspective BP discovery from emails.

- **Introduce an event log structure compatible with multiple BP perspectives discovery:** This structure allows the recording of additional information when activities occurs in emails (**Q1-2**). These information refer to: (i) the exact position of appearance of activities in emails, (ii) activities business information useful for deducing the manipulated artifacts, and (iii) emails senders purposes when expressing activities for inferring actors' contributions and the actual events appearing in emails.

- **Introduce an automated approach for multi-perspectives BP fragments discovery from emails (Q2, Principle 2)**: The introduced approach (i) is totally unsupervised and (ii) does not require a prior knowledge about BP or exhaustive human intervention (**Q2-1**). Our approach transforms email log data into structured event log, and then, it mines it to discover multiple BP perspectives (**Q2-6**). To this end, it combines several algorithmic solutions for:

  - Activity discovery based on pattern discovery from emails: This solution discovers frequent activities from emails. This includes the automatic generation of their names and their related business information while considerably reducing human intervention (**Principle 2**). We rely on the discovery of frequent activities as a first step in our approach as they could potentially characterize BP. For instance performing BP according to different instances leads to repetitively executing the same set of activities. We discover each activity name or business information in the form of a set of patterns reflecting the set of combination of words frequently used by employees to express it in emails. We suppose for this purpose that if the same employee frequently expresses the same activity component in his/her emails, he/she would use close expressions. With this manner, granularity expression levels are defined according to the ones granted by BP actors, i.e. according to what they write (**Q2-3**). Additionally, this allows the discovery of multiple BP elements (i.e. activity names and business information) per one email (**Q2-4**) as one email could contains multiple patterns referring to different BP elements. Furthermore, we rely in this way on the number of occurrence of the combinations of words used by employees in emails to make a trade-off between their degree of significance and the recall of the discovered BP knowledge (**Q2-2, Principle 4**).

In order to reduce the variance of the writing styles of different employees, we analyze only sent emails per employee. Then, similar activities of different employees are grouped.

- Activity actor perspective discovery based on predicting the sender purpose when expressing activities in emails. This purpose is predicted by discovering senders' speech acts at the level of each expressed activity which is useful for inferring actors contributions.
- Overlapping clustering of BP elements for discovering BP fragments (i.e. functional perspective) and artifacts manipulated by activities (i.e. data perspective): This enables designating one BP element to more than one BP fragment/artifact, which better reflects the reality of activity organization over BP and artifacts manipulated by them.
- BP behavioral perspective discovery based on mining sequencing constraints between activity events and events combining activities and their speech acts. We used activity speech acts to estimate the actual relative sequencing order of activity events (**Q2-5**). To describe the behavioral perspective models, we discover sequencing constraints between event types to provide more flexibility for modelling BP with high variability (which is the case of employee-driven processes executed through emails).

We implement a framework for our overall contributions including a visualization tool for multi-perspective BP discovery; for each BP perspective, we automate the generation of its corresponding graphs enabling users to visualize and interact with the discovered results. Finally, we validate all the introduced algorithmic solutions using real emails belonging to the public Enron dataset and/or emails of Orange employees (**Principle 3**); we publicly provide our experimental results (see this link[2]) to make quantitative comparisons feasible with related studies if they use the same public dataset. This allows more practical analysis for further research.

We finally show the usefulness of our results for improving BPM in less structured IS. We present two potential applications: (i) a BP discovery & recommendation tool to be integrated in emailing systems, and (ii) CRM (Customer Relationship Management) data analysis for mining reasons of users' satisfaction/non-satisfaction.

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 presents a state of the art analysis related to our work. First, it presents an overview on the BP discovery from IS execution logs, which corresponds to the general context of our study. Then, it presents a more thorough analysis related to existing approaches that discover BP knowledge from emails. Finally, it discusses and compares these approaches and highlights the research questions that are not studied to

[2]http://www-inf.it-sudparis.eu/SIMBAD/tools/processDiscoveryFromEmails/

date. This analysis allows us to justify the need for proposing a totally unsupervised approach for discovering BP fragments from emails w.r.t multiple perspectives.

Chapters 3, 4, 5, 6 and 7 are the core of our thesis which elaborate our main contributions. Chapter 4 presents an overview on the overall approach that we propose for BP fragment discovery from emails w.r.t multiple perspectives. Chapter 4 presents its related metamodel and formalizes the definitions of its main entities. The next three chapters (Chapter 5, Chapter 6 and Chapter 7) detail the algorithmic solutions that we introduce for ensuring the main phases in our approach, i.e. event log generation and event log mining. For instance, Chapter 5 presents the unsupervised learning solution for activity discovery from emails, which ensures a first and preliminary step towards event log generation. Chapter 6 continues to present the remaining steps towards obtaining a structured event log from emails. Finally, Chapter 7 detail the techniques that we introduced form mining BP fragments from the generated event log w.r.t multiple perspectives.

Chapter 8 shows how our introduced algorithms would be applicable and extended in other contexts. Additionally, it shows how they would open the door for smart functions in emailing systems and BPM. Tow potential applications are presented: (i) Analysing verbatim records of service users for mining satisfaction/non-satisfaction reasons, and (ii) BP discovery & recommendation system to be integrated in emailing systems.

Finally, Chapter 9 concludes this thesis by summarizing the work presented and discussing possible perspectives.

Chapter 2

State Of The Art

Contents

2.1 Introduction

In this chapter, we review the existing studies relevant to the topic of BP discovery from emails. We identify two main research areas in relation to our study. The first one represents the global context of our research, which corresponds to BP discovery from IS execution logs. The second one is the research area that is directly related to our study, which corresponds to BP knowledge discovery from emails. In Section 2.2, we present an overview on the first

research area. We outline the main approaches that are generally used for preprocessing IS execution logs and the main algorithms commonly used for BP discovery purpose. In the sections that follow, we detail the second research area which is directly related to our work. In Section 2.3, we outline the methodology we adopted for selecting existing studies that discover BP knowledge from emails and we hierarchically categorize them. We distinguish two main categories defined on the basis of the type of BP knowledge discovered by each existing work: (i) Activity discovery approaches, and (ii) BP discovery approaches. Section 2.4 and Section 2.5 detail these two categories. Finally, Section 2.6 discusses and compares existing studies belonging to the second research area. Since no implementation is available concerning them to perform comparative analysis on a technical and quantitative basis, a qualitative evaluation is considered. This evaluation aims to compare what a proposed approach allows doing with what would be qualitatively allowed according to a set of evaluation criteria. We thus propose a set of evaluation criteria in Section 2.6.1, and then confront the existing studies to these criteria in Section 2.6.2.

The work presented in this chapter (mainly in Section 2.3, Section 2.4, Section 2.5 and Section 2.6) was published in the international conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) [30] and the Concurrency and Computation: Practice and Experience journal [32].

2.2 BP discovery from IS execution logs

Our work is globally related to the area of process discovery from IS execution logs. In Section 2.2.1, we start by outlining how to assess the quality of an execution log from BP discovery perspective. We additionally resume the main categories of these execution logs. Then, in Section 2.2.2, we give an overview on existing studies that preprocess them for BP discovery purpose. In Section 2.2.3, we outline the main algorithms commonly used for process discovery from the preprocessed execution logs. Finally, in Section 2.2.4, we discuss the quality of emailing system execution logs from BP discovery perspective. We additionally resume the main points making BP discovery from emailing systems' logs different from BP discovery using execution logs of other IS.

2.2.1 Execution logs categories

From BP discovery perspective, three criteria are mainly considered to judge the quality of IS execution logs according to the process mining manifesto [91]:

(i) Trustworthiness: The events should be safe to assume that the recorded events actually happened and that the attributes of events are correct;

(ii) Completeness: The event logs should be complete, i.e., given a particular scope, no events may be missing;

(iii) Safety: The event data should be safe in the sense that privacy and security concerns are addressed when recording the events. For example, actors should be aware of the kind of events being recorded and the way they are used.

Based on these criteria, we could differentiate between three categories of IS execution logs [13]:

(i) Execution logs of high level of maturity: They correspond to execution logs whose events' recording rely on a systematic and reliable manner. This results in execution logs of high level of trustworthiness, completeness and safety, where BP elements of each event could be easily inferred;

(ii) Execution logs of intermediate level of maturity: They correspond to execution logs whose recording do not rely on systematic approach to ensure their inner-coherence and their compatibility with BP event log structure. They could be resulted from using different IS, in the context of the same BP, but without these IS being aware of the BP models followed by BP actors when performing activities. In other words, each one of these IS could support one or multiple basic functionality but without being configured to take part in a defined BP. In such situation, orchestration between the operation of these IS is not necessary automated, it could be driven by workers implicit knowledge. This results in obtaining execution logs where BP elements or at least some of them (e.g. BP instance), are not necessary recorded in an explicit manner or in a uniform way. Therefore, some preliminary steps to ensure BP discovery could not be directly performed (e.g. map events handling the same BP instance or the same BP). This results in the need of a preporcessing step to extract BP elements from such execution logs to transform them into logs of higher level of maturity;

(iii) Execution logs of low level of maturity: They correspond to execution logs of poor quality whose recording do not rely on a systematic approach to decide which events are recorded. Therefore, events may be missing or not recorded properly. They could be recorded not only automatically (e.g. error logs of embedded systems) but also manually (e.g. paper-based medical records). It is worthy to note, that the same needs mentioned at the level of execution logs of intermediate level of maturity, are also present with that of low level of maturity. However, further preprocessing steps are required to be implemented. Sometimes, human intervention is additionally necessary to digitize the manually recorded execution logs;

2.2.2 Execution logs preprocessing for BP discovery

Process discovery techniques could be directly applied on execution logs of high level of maturity. This kind of logs are known in the litterature as *'labeled event logs'* [91]. However, starting from execution logs of lower level of maturity calls for a preprocessing step that addresses the challenge of « Finding, Merging, and cleaning Event Data » [91]. This challenge is concerned with extracting BP elements from execution logs that could be distributed over

different sources (i.e. logs repisotory of different non process aware IS). The aim is to extract and prepare event logs to be compatible with BP discovery. The most common problematic that was handled by existing studies is when different instance identifiers are used by the different IS taking part in the same BP. For example, one system uses name and birthday to identify a person whereas another system uses the person's social security number. Execution logs with no common instance identifiers enabling correlating events handling the same instance are called in the literature as *'unlabeled log data'*.

Several approaches have been proposed to repair these incomplete and unlabeled event logs and to convert them into labeled ones. Rogge et. al associate to a given BP a stochastic model [79, 78]. Then, they use its path probabilities to determine probable missing events in each trace or/and their probable timestamps values.
Some studies [27, 97, 14, 99, 67] preprocessed web services interaction logs with the aim of discovering the related BP (i.e. interactions' workflow). The related execution logs are considered of low quality as the instance identifier is absent. Such BP element was supposed to correspond to web service sessions. To identify these sessions, some studies adopted a time based-approach [97, 99, 27]. They assume that: (i) execution logs are related to the same BP, and (ii) sessions have a duration threshold, i.e. the time lapse between two successive interactions that marks the end of one session and the beginning of another. The duration threshold was considered as a constant value in some studies [97] and flexible and adjustable by others [99]. For Dustdar et al. [27], the duration threshold was continuously updated until finding sessions of good quality. Such quality is assessed based on the assumption that sessions should be similar in terms of duration, number of interactions, consumed services in a session and their relative order.

Other studies have dealt differently with the problem of instance ID discovery. They considered it as an *'event correlation task'*. Motahari-Nezhad et al. [67] introduced in this context two notions: (i) a set of correlation conditions that regroup events belonging to the same instance, each atomic condition specifies that two events are correlated if they have the same value on two of their attributes, and (ii) process view that refers to the process resulting from a specific way of correlating events based on a given correlation conditions. The notion of process view was introduced to deal with the subjectivity of event correlation conditions defining one instance. Indeed, the same set of events belonging to the same BP instance may be seen to be related to different ones depending on the application, business domain and person interest. E. G. L. de Murillas et al. [69] presented a framework where they recommend process views to the user and rank them by interests. They defined a *complex case notion* concept to be able to adopt different process views. A case notion was introduced as an annotated rooted tree. Each node in the tree refers to an atomic instance ID (e.g. instance ID of a purchase order). Nodes linking to it refers to the instance IDs performed in its context (e.g. several instances IDs of deliveries of the same purchase order). In this way, the maximal set of combination of instance IDs defining process views are generated from directly or transitively connected nodes while verifying certain constraints. Afterwards, different logs are built according to each process view while assessing their interestingness quality (i.e. to what extent a log is interesting to be analysed). Such quality is deduced from: the number of traces present in an event log and the average numbers of unique activities and

events per trace. Finally, the result of this assessment is used to provide recommendations to the user.

To store the preprocessed execution logs in a suitable format for BP discovery, several standards were introduced to define metamodels describing event logs storage structure. XES (i.e. eXtensible Event Stream) is the most adopted one. It has been accepted as the IEEE standard in 2014 for the storage of the most common event logs [95]. However, such standard allows capturing and correlating events w.r.t a fixed and clearly defined notion of a BP instance (i.e. one process view). Artifact-centric/ Object-centring event logs were introduced to overcome such limitation. They focus more on capturing informational entities related to data perspective and they use artifacts to correlate events. Three metamodels/standards were proposed to store artifacy-centric event logs: OpenSlex [70], eXtensible Object-Centric (XOC) [57] and OCEL [35] (Objecti-Centric Event Logs).

2.2.3 BP discovery from preprocessed execution logs

Once preprocessing execution logs and transforming them into event logs of higher maturity, a BP discovery algorithm is applied to discover how BP are actually performed. According to a recent study [20], ProM [94] and Disco [36] are the tools that are commonly used for BP discovery.

Most often, existing studies rely on these tools to discover the behavioral perspective of BP according to two main types of BP models: imperative and declarative. The imperative models describe the exact sequence of activities that should be done in the context of BP. They are compatible with highly structured BP. Existing approaches most often apply the following algorithms to discover BP models w.r.t imperative descriptions: Fuzzy miner [37], heuristic miner [96], genetic miner [3] and α-algorithm [1]. The declarative models describe BP through a set of constraints defined over the set of activities. They are compatible with less structured BP. In fact, they specify what are the next allowed activities at each stage of the BP execution rather than the overall sequence of activities that must be performed. DECLARE was introduced as a declarative BP modelling language [92]. It defines a set of constraint types such as *Response*, *CoExistence* and *Precedence*. Among the techniques allowing BP discovery w.r.t to DECLARE language, we cite MINERful [23], which is a fast scalable plugin integrated in ProM.

BP discovery algorithms generally regroups events per BP trace using the instance ID information. Then, they are commonly based on timestamps' information for ordering events in each trace. Afterwards, they mine, from the obtained traces, frequent and reliable patterns of successive and dependent event pairs to construct BP models.

A good part of existing BP discovery algorithms assume a *single* and *explicitly defined notion of a BP instance*. To allow BP discovery w.r.t different views resulted from the combination of multiple instance IDs, we cite the artifact-centric process mining approaches (a.k.a object-centric process) [2]. These approaches starts from the artifact-centric event logs to allow the discovery of BP of w.r.t multiple instance notions. For example, in a recruitment

process, one can consider the *'application'* and the *'vacancy'* as the BP instance notions for different parts of the BP. They additionally allow the discovery of the artifacts manipulated by BP activities, which describes BP data perspective.

When discovering the organizational perspective of BP, additional techniques (usually ProM plug-ins) are used such as Organization miner, Social Network Miner and Handover Of Work algorithm. These algorithms mainly describes performers of each BP activity/case and the relations among BP activities' performers (or groups of performers). They generally suppose that, for one activity instance, it has one performer. Nevertheless, several actors can actually intervene to perform an activity. Such actors would not be all performers; they can collaborate with different collaboration types so that the activity would be performed. A recent work [81] discovered different actors involved in the execution of an activity. However, it is supposed that an activity is represented and implemented as a sub-process and actors are performers of this sub-process. Therefore, interactions with other collaborators (that are not necessarily tasks' performers) remain missed.

2.2.4 Discussion: Execution logs of emailing systems from BP discovery perspective

From BP discovery perspective, email logs could be viewed as logs of low level of maturity. On the one hand, they record events referring to emailing systems operation mode (e.g. sending, receiving emails) rather than BP oriented events. Additionally, as explained previously, not all activities reported in emails have actually happened. In fact, they could be reported with various purposes such as request, request information intention or information of execution. This means that email logs are of low level of trustworthiness. On the other hand, there is no guaranty that all BP events w.r.t each instance are reported in emails, which means that email logs are of low level of completeness.

Previously cited studies suppose that IS execution logs record BP enactment. This means that each event is related to the execution of one BP activity. That's why, they assume that some BP elements (e.g. activity names) are to be directly inferred from those of the execution logs' events or from database tables where execution logs are stored (e.g. artifacts and their interconnections are inferred from tables and their primary and foreign keys). Actually, this is not the case of email logs. Email logs could be viewed as logs capturing actors' interactions and conversations around BP activities rather than only capturing their execution trace. Additionally, if a BP element appears in an email, it is expressed in natural language. This requires further prepossessing steps to extract it.

For ensuring BP discovery according to the behavioral perspective, commonly used algorithms rely on explicit timestamp information of events, which is absent in the case of emails. As for the organizational perspective discovery, existing algorithms mainly suppose that for one activity instance, there is one performer. Nevertheless, several actors can actually intervene to perform an activity. Such actors would not be all performers; they can collaborate with different collaboration types so that the activity is performed. Relying on traditional

execution logs, only the trace of the activity performers is recorded. Interactions that may be happened before or after execution would be missed.

To summarize, BP discovery from emails differs from BP discovery using the previously discussed execution logs due to two main reasons: (i) the former requires further preprocessing steps to structure emails into structured event logs, (ii) emails have different specifities comparing to execution logs recording BP activity enactment (e.g. the expression of BP elements in natural language, the possibility of using emails in the context of BP activities for different purposes, etc). In what follows, we will focus on existing studies that analysed emails to discover BP knowledge.

2.3 BP knowledge discovery from emails: Research Methodology & Existing studies' categorization

This section presents the research methodology we adopted for selecting related studies to BP knowledge discovery from emails (see Section 2.3.1). Then, it presents our proposed categorization of these studies (see Section 2.3.2) that we further detail in Section 2.4 and Section 2.5 and that we discuss in Section 2.6.

2.3.1 Research Methodology

To define the current state of research and to identify different approaches for discovering BP elements from emails, a systematical literature review was conducted. Literature databases such as Google Scholar, SpringerLink and ResearchGate were used for the research. The following combination of keywords were mainly used: BP/activity/ worker knowledge mining/discovery from emails, email/email analysis for activity/process discovery/mining. As these used keywords could diverge for other research areas (e.g. in relation to text mining, network analysis from emails, BP discovery from structured logs data, etc), the obtained studies were checked through their titles and abstracts to ensure that they verify the following elements:

- Start from a log data of messaging systems, e.g. chats, emails, forum discussions, etc;
- Target the discovery of complete/partial BP or at least one BP element (e.g. activity, instance);
- Introduce an algorithmic solution or a benchmark dataset in the context of discovering BP from emails;

Afterwards, referred studies in the selected papers were also checked in an iterative and backward way to enrich our research results. Actually, due to the lack of studies that discovered BP discovery from emails, we were not based on the number of citations criterion. Nevertheless, this does not guarantee the exhaustiveness of our results.

2.3.2 Categorization

In the context of BP discovery from email log data, existing studies could be organized according to three categorization levels, as illustrated in Figure 2.1 (the number of identified papers in each category/sub-category is indicated in parentheses after its name). In the first level, we divide them into two main categories according to the business knowledge type that they discover. One of these categories discovers activities without grouping them into BP. The other discovers BP consisting of both: a list of activities and a list of associated sequencing constraints (most often visualized in the form of BP models). In the second level, we divide existing approaches according to: (i) the multiplicity of activities allowed to be discovered per one email (i.e. Single activity and multiple activities per one email approaches), and (ii) the employed techniques (i.e. Rules based and learning based approaches). The last categorization level further details either the employed techniques or the type of the input data (i.e. IS driven communication data or not).

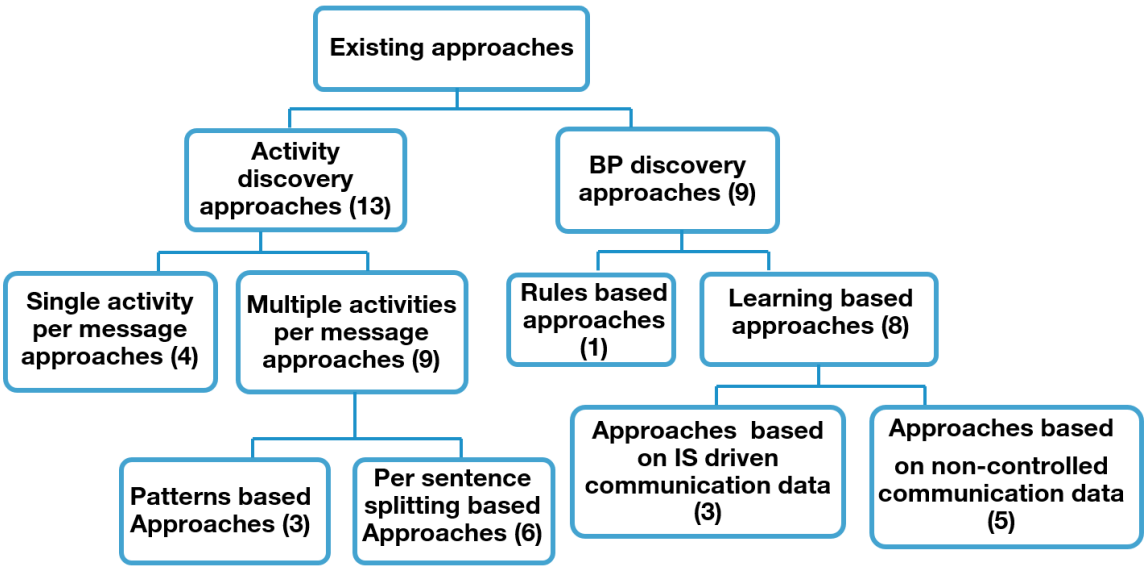


Figure 2.1: Hierarchical categorization of existing studies

2.4 Activity discovery approaches

This section focuses on the first category of existing approaches belonging to the first categorization level (See Figure 2.1). These approaches only discover activities without organizing them into BP or mining their behavioral perspective. According to the number of activity allowed to be detected per email, approaches in this category are divided into two sub-categories: The first one regroups the approaches that enable the discovery of a single activity per one email [25, 50, 49, 65]. It consists of assigning one label to each email using supervised [25], semi-supervised [50, 49] or unsupervised learning techniques [65]. The second sub-category regroups the approaches that enable the discovery of multiple activities per one email [54, 18,

76, 77, 43, 86, 48, 29, 31].

2.4.1 Single activity per email approaches

These approaches allow the discovery of one activity per one email. This consists in assigning a single activity label for each email. Dredze et al.[25] introduced an incremental learning approach that transforms the problem of assigning one activity to each incoming email into a recommendation problem. To this end, they proposed three recommendation algorithms: SimContent, SimSubset and Threading. Then, on the basis on their results, they used a voting approach to select the activity that best matches the concerned email. These algorithms calculate scores that evaluate how probable an incoming email is related to an activity by computing a similarity metric between the email and the known activity emails. These scores are then used to generate the top N activities that best match the concerned email. According to each recommendation algorithm, the similarity score with each activity are obtained as follows: (1) SimContent: by adding up the similarities of incoming email with all existing emails having the same activity label, (2) SimSubset : by calculating the fraction of people in the incoming email that belong to each activity, and (3) Threading : by considering incoming email history; if it is a reply to a previous email, the activity to which the previous email belonged will be the most similar, otherwise it recommends the null activity.

This incremental learning approach [25] can reduce individual human effort when annotating training data. It provides also an interactive graphical interface to facilitate this annotation task. However, it relies on some assumptions that are not verified in some contexts. In fact, the threading recommendation algorithm supposes that emails belonging to the same thread (origin email + replies) discuss the same activity. Such assumption is not always verified. Indeed, in some contexts, employees may discuss different activities in the same email thread. For instance, Figure 2.2 shows a conversation example extracted from Enron database and which concerns a recruitment process. This conversation starts by an email sent by a candidate to an employee for applying to a job (email in the red rectangle). This employee showed an interest about the received candidature, then, he transferred it to his manager to ask for his opinion (email in the green rectangle). Finally, the manager confirmed his interest about the candidature and requested his assistant 'Shirely' to schedule interview with the concerned candidate. As a consequence, each email in this thread refers to an activity which is different from the previous one. Our investigation of Enron and Orange email database confirmed that this case is common.

Khoussainov et al. [50, 49] proposed an iterative relational learning approach for email task management. This approach exploits the mutual performance improvement between the extraction of speech acts and the identification of related emails. The first step consists of initializing both of them using automatic methods; it identifies related emails using content, subject and time similarities and it classifies emails into speech acts using only the email content. In the second step, a supervised learning algorithm (SVM (Support Vector Machine) with SMO (Sequential Minimal Optimization) implementation [98]) relies on the results of the first step to train two classifiers that will be applied later on incoming emails. The

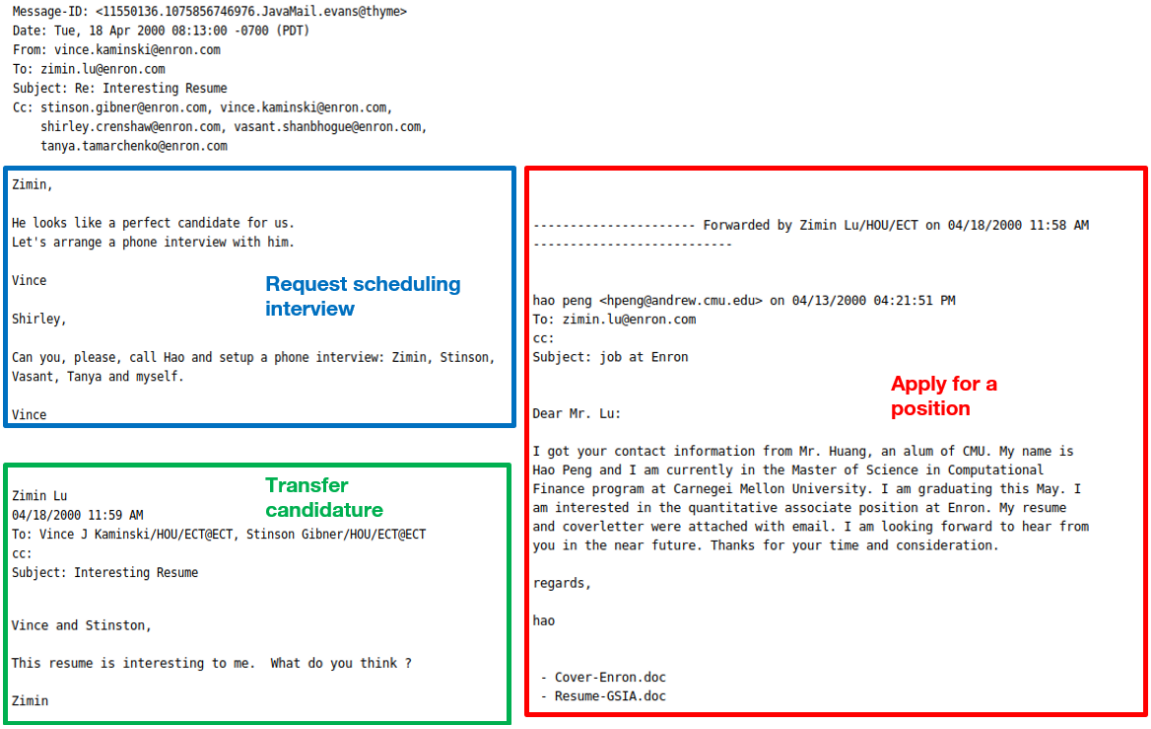


Figure 2.2: Email conversation from Enron dataset

first one identifies if two emails are related. The training examples consist of email pairs represented by their similarity score and binary features that reflect their speech acts. Then, by exploiting the results of initialization step, each pair is labeled as positive if its emails are related and negative otherwise. The second classifier identifies if an email contain or do not contain the predefined speech acts. The examples used in training phase contain emails of initialization step associated with their speech acts and represented by intrinsic (derived from email content) and extrinsic features (derived from related emails and which corresponds to their associated speech acts).

In another work [52], Khoussainov et al. extended their relational learning approach [49] in order to formalize activities as finite state automata [51]; the key idea is that speech acts are viewed as the analogs of activity states. As consequence, the proposed approach contained the same previous steps [49] ; one initialization step to cluster past emails into activities and assign them to speech acts and a second step to handle incoming emails. These approaches [49, 51, 52] allow the tracking of activity states which serves the user in the management of his/her tasks. However, they rely on a supervised technique for speech acts identification [52, 51] which leverages, as initial step, the use of labeled training dataset to train a speech act classifier and which requires human involvement.

Mitchell et al. introduced an approach to automatically discover the descriptions of workstation activities of one user, formalize them into structured format and use them later to provide him with knowledge-based assistance of its main activities [65]. The structured description of one activity includes a list of keywords, emails and meetings associated with it and addresses, names and involvement fractions of its actors. To generate such descriptions,

a three-step algorithm was used:

1. Cluster email threads using an EM-algorithm (i.e. Expectation-Maximization algorithm) to generate initial activity clusters on the basis of bag-of-words representation of its textual data. The number of cluster and their initial vectors (required for the EM-algorithm) were calculated by using a heuristic method or by using an optional input containing activity names;
2. Refine the initial activity clusters (generated by step 1) by using a social network analysis to remove isolated emails that do not belong to them;
3. Create structured descriptions of each activity by deducing them from the emails assigned to it. The description of each activity includes the list of recurrent keywords and the name of employees involved in its execution;

This work have combined social network analysis with email content analysis to discover activities: the idea seems good to detect communities of employees that often collaborate on the execution of one activity or one process. However, it assumes that one worker executes each type of activity with the same community of employees. This assumption is not always true. For instance, the HR team may contact different candidates (which does not belong to the same community) to organize interviews. Additionally, it may contact different recruiters within a company to be informed about the selected candidates. Consequently, the network could not characterize the activity. Another limitation to be mentioned here is that the EM-clustering algorithm used to detect activities requires the definition of the number of clusters as input which is not obvious to know in advance.

2.4.2 Multiple activities per email approaches

To enable the discovery of multiple activities per email, proposed approaches relies on: (1) splitting emails into sentences and then, assigning these sentences to activities (using supervised or unsupervised learning techniques)[18, 43, 76, 77, 86, 48], or (2) associating to each activity a set of patterns of words enabling its detection in emails without being constrained by emails or sentences structures [54, 29, 31].
Sentence based approaches: These approaches suppose that one activity is expressed at a sentence level. In Figure 2.3, we illustrate the main steps adopted by them to discover activities from emails. First, they split each email into sentences. Then, each sentence is assigned to an activity type using supervised learning approaches [18, 43, 76, 77], pattern based approaches[86] or clustering techniques [48]. Activity discovery from sentences is, in some studies, preceded by another step carried out manually [86] or with supervised learning methods [48], and which aims to identify relevant sentences that may express activities.

Most of sentence based approaches deal with activity name recognition by using speech act theory based methods: The idea behind this theory is to classify email sentences according to the sender's intent [18, 76, 43] or to use these intents as key passage to deduce activities [77]. In the literature, three possible classifications of speech acts are proposed:

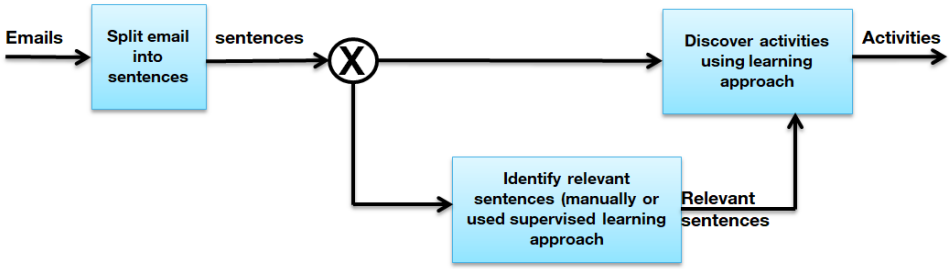


Figure 2.3: Main Steps for discovering multiple activities per email

- Illocutionary act classes [82]: assertive, commissive, directive, expressive, declarations, etc;
- Speech act verbs [83]: propose, request, deliver, commit, etc;
- Emails-oriented speech acts [77]: salutation, chit-chat, meeting, promise, farewell,email signature, etc;

Some proposals define the list of pertinent speech acts in advance. Then, they apply a supervised learning algorithm [18, 76, 77] or semi-supervised learning algorithm [43] to classify email sentences as containing or not containing the specific acts. SmartMail [77] introduces an additional step to reformulate act-oriented sentences, and present them to the user in a convenient interface to enable them to easily populate their to-do list. Obviously, such studies require labeled data for training statistical speech acts recognizers which leads to a significant human intervention. Furthermore, BP tasks differ from one BP to another. Thus, setting a unique list of activities in advance, degrades the performances of generating the right BP models.

Sentence based approaches also includes studies that discovered some metadata correlated to activities during their execution [86, 48]. E-Mail Mining [86] introduced a method for semi-automatic discovery of knowledge-intensive process. From a set of emails belonging to a BP, e-Mail Mining aims to discover activity knowledge consisting of :

- BP participants and their social interactions;
- Relevant terms that are related to the BP domain;
- BP activities defined by three elements : Actors, candidate actions and parameters

Relevant BP activities are selected manually from a list of candidate activities and their embedded knowledge are detected by assuming that each sentence is composed of:

- A noun phrase object which describes the agent performing an action or the resource that receives the effect of the executed action;
- A verb phrase that describes the activity performed by the agent;

Jlailaty et al. has addressed the same aspect [48] as Soares et al. [86] but with different approach. To discover BP activity types, a hierarchical clustering is applied on all email

sentences after identifying their verb-noun pairs and filtering them by keeping only those oriented to BP activities. To identify this latter type of relevant sentences, a supervised learning algorithm (Gradient Boosting Classifier) is used in order to train a binary classifier (relevant / irrelevant sentence). The clustering phase is based on a similarity measure between relevant sentence pairs. This measure is computed using the cosine similarity between the word2vec vectors of their verb-noun pairs. In order to extract activity metadata, Jlailaty et al.[48] extract information related to each activity type instance and then, aggregate them to get the metadata describing the corresponding activity types. The metadata can be: Organizational role of the people exchanging the email, actors that are actually performing the activity described in the email, attachment types and email-included web pages descriptions.

One of the main limitations of such approaches [48, 86] is that they require human intervention in some of their key steps. In fact, Jlailaty et al. use supervised approach to select relevant sentences which implies the need of human involvement for collecting and labeling training data [48]. As for E-mail Mining [86], it requires manual tasks during its execution (e.g for selecting relevant activities or sample of emails related to one BP). The second limitation of such approaches is to rely on some algorithmic assumptions that seem to be limited. In fact, E-mail Mining [86] considers emails as storytelling textual data to mine the candidate activities. Actually, emails do not have the same structure as narrative textual data (which generally describes activities in a more formal way than e-mails). For instance, the proposal does not seem to handle passive-voice sentences where actors do not appear or where their positions are switched with those of resources. Additionally, Jlailaty et al. use word2vect to cluster activities [48]. Actually, word2vect groups terms having the same context without differentiating between their meanings. This would create some ambiguity in the context of activities discovery. Taking the example of two antonyms: *'buy'* and *'sell'*, they are generally grouped together when using their word2vec representations, although they would refer to two different activities; buy a transaction and sell a transaction.

Patterns based approaches: These studies use the notion of patterns of words to facilitate activity detection without being constrained by emails or sentences structure. This means that even in one sentence, multiple activities could be discovered. Laga et al. [54] assumed that a manual task is an association of (1) BPMN2.0 (BP Management Notations 2.0) specification [66], and (2) a set of semantic patterns that enable to validate whether a given communication content is part of a given BP activity. However, this requires anticipating and manually defining all semantic patterns related to each task.

2.5 BP discovery approaches

This category discovers BP in the form of: a list of activities and a list of associated sequential constraints (most often visualized in the form of BP models). The common BP elements discovered in each approach are: BP names, BP activities, BP instances and constraints related to BP activities sequencing. These approaches consist generally of assigning to each email one BP name, one BP activity and the corresponding BP instances. Then, the email logs will be converted into event logs where each event is characterized by a timestamp, the

BP name and the executed activity name. Events are then grouped according to the BP to which they belong, and for each BP events group, events are grouped according to their instance. Finally, activity sequencing constraints of each BP are deduced.

Existing approaches in this context differ mainly in the adopted techniques to discover BP elements. As illustrated in Figure 2.1, some approaches rely on manually defining rules to assign BP elements to a given email [4]. Some others integrate learning techniques to partially or totally automate the definition of such rules [44, 46, 24, 22, 55, 11, 51, 52].

2.5.1 Rules based approaches

This category is based on manually defining rules that assign each type of BP elements to emails. One of these rules consists of specifying, in which kind of textual pattern, one information type can appear in an email. Van Der et al. for example assume that the associated BP name is explicitly included in the email subject [4]. Such an approach supposes that email interlocutors must include the BP name and the related BP elements in the email subject. Such assumption is not always true. Taking the example of the emails in Figure 1.1 and Figure 1.3, BP name and instance are absent from email subjects.

2.5.2 Learning based approaches (supervised & unsupervised)

Learning based approaches [44, 46, 24, 22, 55, 11, 51, 52] were introduced in order to automatize assigning BP information to emails. Generally, such approaches integrate supervised, unsupervised techniques or both of them. One of the advantages of integrating them is to decrease human intervention in defining algorithmic rules (vs. rules matching based approaches). Moreover, unsupervised learning techniques are useful in the case of not disposing a prior knowledge about some or all the BP information to be discovered. They can be also useful when aiming to avoid preparing a labeled dataset that requires human effort. In what follows, an overview of existing studies having used learning techniques will be introduced. They will be discussed according to the type of input data: IS driven communication data and Non-controlled communication data.

- IS driven communication data: These approaches suppose that communication data are driven by an IS, which implies that for each communication trace, we dispose some BP elements such as the name of the BP or the ID of the BP instance (or both of them) [11, 51, 52]. This kind of studies completes the missing part of some BP partially executed in some IS with the assistance of messages. However, it could not be generalized to the absence of a regular IS. In fact, there will be more variance in the human-driven BP and BP elements would not be provided in standard format.

Banziger et Al.[11] mined BP from CRM driven communications while supposing that BP names and cases IDs are known for each communication trace. For this purpose, emails of the same BP are grouped according to their direction (sent or received). Then, an LDA (Latent

Dirichlet Allocation) model is fitted to obtain a combination of N topics. N is initialized with the average number of activities per case. To obtain the activity from a textual data of one interaction type, the corresponding LDA model is applied and the five most probable words of the most likely topic is returned.

Kushmerick et Al. started from e-commerce transactions (of a single e-commerce vendor) [51] for discovering BP. It suggests formalizing them as finite state automata where: (1) states correspond to activities not executed through emails as those that are executed through e-commerce sites (e.g. "order a product") and those that are executed by vendors (e.g. "deliver a product") and, (2) transitions correspond to emails ensuring the passage from one state (activity) to another. To automatically generate these automata, an unsupervised approach is applied consisting mainly of: (1) Identifying emails of the same transaction instance of a given vendor by clustering emails on the basis of their alphanumeric sequences (that can correspond to order numbers, item numbers...) and, (2) Identifying transitions between activity states; the idea is to suppose that email contents indicate the completion of one activity or the beginning of the following activity (or both of them), which means a transition from an activity to another. Such kind of emails are likely to contain common templates. That's why, emails are clustered on the basis of the long common sub-sequences shared between their textual contents (e.g. all order confirmation emails tend to contain the same text thanking the user for having placed an order). This work allows the tracking of activity states which serves the user in the management of his tasks. Nevertheless, some steps in the proposed approach do not seem to be clearly explained; how activity states are discovered from the discovered BP? And how these states as well as their transitions are annotated?

- Non-controlled communication data: Approaches starting from such data aim to discover BP in the absence of a regular IS, which means that BP are driven by employees.

Jlailaty et al. proposed two studies: one study for discovering BP and activities from emails [46] and another study for discovering BP instances [45]. To discover BP and activities, a hierarchical clustering is sequentially applied to deduce BP clusters and then sub-clusters corresponding to activities. To measure similarities between each couple of emails when applying the clustering algorithm, the proposal integrates word embedding method (word2vect model of Google); each email is represented by the average sum of its words vectors (belonging to email content and subject), then, a cosine similarity is applied between each couple of email vectors. An activity labeling technique is also proposed: It recommends to the user the most occurring contiguous sequence of n items in an activity cluster. To discover BP instances, the authors discover first BP [46]. Then, for each BP group of emails, a hierarchical clustering algorithm is applied. The distance matrix used in the clustering phase is defined as a weighted sum of sub-distances related to the following email parts: (1) time, (2) email participants, and (3) content and subject reduced into named entities. Each weight correlated to each sub-distance is tuned experimentally according to the related BP.
Jlaitaty et al extended, in another work [47], their approach that allows the discovery of multiple activities per email. They combined it with their BP instance approach [45] to build an event logs compatible with the discovery of BP considering the multiplicity of activities per one email. To this end, they extracted temporal expressions from emails to identify the order

of execution of activities appearing in the same email. Here, two temporal relation types were identified : (1) between the email activities and the email timestamp, and (2) Between the email activities themselves.

These studies [46, 45] discover BP elements without disposing a prior knowledge. However, they are based on the use of a hierarchical clustering algorithm which requires a human effort in tuning its parameters. As consequence, applying the same algorithm twice in the same method increases the degree of human intervention [46, 45]. Moreover, using word2vect for clustering emails into BP seems to remove a part of the available information in emails (which must contribute in separating them into BP). In fact, the used google model of word2vect is likely to not know the abbreviations and technical terms used by employees and which differ from a company to another.

Other approaches combine classification and clustering techniques for discovering BP models from emails [55, 7]. The first approach of Laga et al. [55] initially ensures the step of BP and activity labels generation manually and collaboratively by users. Then, a predictive model is trained gradually with the obtained annotated data for recognizing BP and activity names. The classification features, which are used in the training phase, are built from the following email parts: subject, content, historical exchange and correspondent entities of email interlocutors. Once reaching reliable prediction performances, the generation of BP and activity labels will be automatically performed by the obtained predictive models. As for BP instance detection, a clustering algorithm HDBSCAN [64] is applied. The used distance matrix is defined as a weighted sum of sub-distances related to the following email parts: (1) time, (2) correspondent entities of email participants, and (3) content and subject reduced into references and named entities. Each weight correlated to each sub-distance is tuned experimentally according to the related BP. This work proposes a collaborative and iterative approach which is implemented through graphical interfaces. This has the advantage of encouraging users to be involved in building an annotated dataset and to reduce their effort at individual level. However, this approach relies on human involvement. This latter does not only concern the training dataset annotation but also the tuning of some parameters (such as the weights related to the sub-distances composing the similarity measure used in the clustering algorithm). Moreover, tagging collaboratively the same dataset can lead to dispose samples belonging to the same cluster but with different annotations. Therefore, tag normalization step is required.

The second approach of Chambers et al. is mainly composed of three phases [7]; (1) Email analysis, (2) Event logs construction, and (3) Process Modelling. In the first phase of email analysis, activities are associated to emails using a supervised learning approach. Additionally, emails topics were discovered by clustering emails textual content using HDBSCAN algorithm. In the second phase of event logs construction, emails are grouped by instance (i.e. case). First, they are grouped by their subject under the condition of having at least one interlocutor in common. Here emails are likely to be organized according to a set of conversations. Second, dominant topic of each conversation is identified. Finally, conversations are merged if they share: (i) the same dominant topic, (ii) at least one common interlocutor, and (iii) a predefined time interval. This last operation is iteratively repeated until no possible new merges could

be performed. This approach, compared to the previous one [55], requires less classification steps (only this that classifies emails into activities). This means that less human intervention is required. However this do not deny that such kind of intervention remains not obvious for labeling a training dataset to learn an activity classifier. Additionally, some assumptions were adopted (even by Laga et al. [55]) that are not realistic, which concern: (i) the multiplicity of the activity per one email, and (ii) the multiplicity of instance per one conversation (one activity/instance was supposed per one email/conversation).

Other studies treat the problem of BP detection as a problem of conversation finder [60, 24, 22]. Mavaddat et al. [60] suggests firstly classifying emails into BP and non BP related. The business-oriented email emails are then grouped into threads to detect conversations using a refined version of Vector Space Model and a semantic similarity measure. Finally, the interactions in each conversation are labeled by applying the classification of illocutionary acts [83].

MailOfMine [24, 22] mines artful BP in the form of *'declarative'* models. It suggests starting from some assumptions that relates emails and BP structures: (1) Each conversation

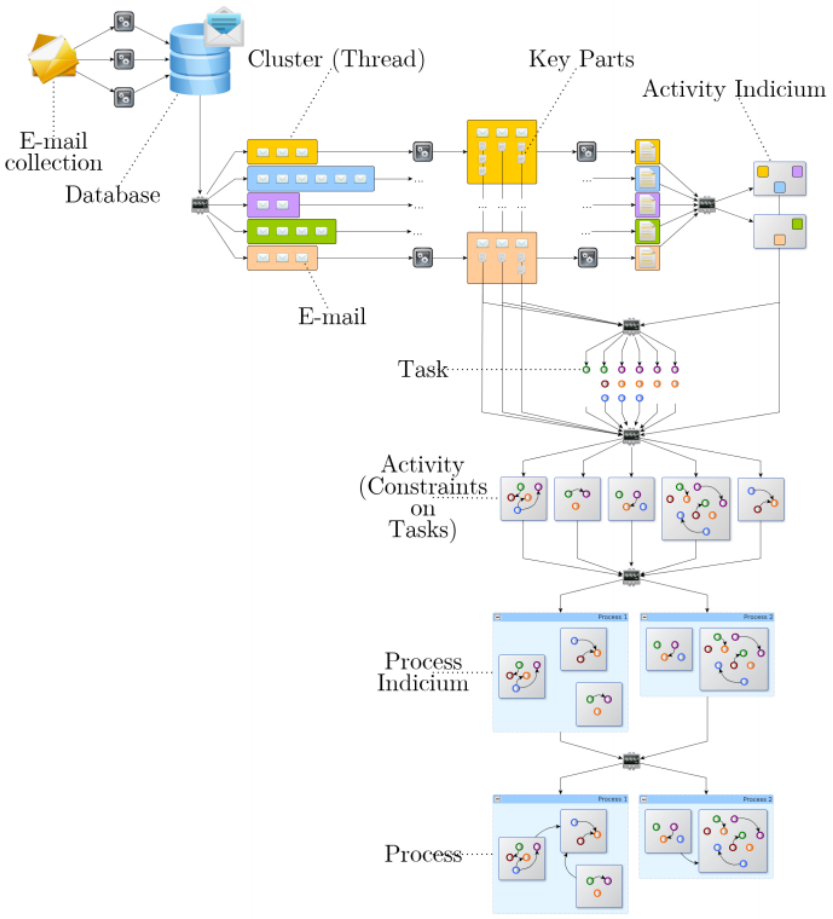


Figure 2.4: The MailOfMine approach [24, 22]

presents an activity trace (2) Each activity presents a set of elementary tasks deduced from conversation key parts (3) Each BP is composed of a set of activities. As depicted in Figure 2.4, MailOfMine consists basically of: (1) Applying three times a similarity clustering algorithm: to cluster emails into conversation threads, to cluster these threads into activities and finally, to cluster each activity key parts into tasks. During the clustering process, email body, the names of attached files and some email header fields are taken into consideration (2) Applying supervised learning process to assign activities to different BP (3) Automatically labeling activity tasks with the possibility of customizing them and manually assigning activity and BP names (4) Mining tasks/activities constraints among each activity/BP. This work discovers BP with different levels of granularity (BP, subprocess or activity, task) and describes them in a declarative way, which is more flexible than the classical imperative one. Nevertheless, a considerable human intervention would be required to manually assign activity and BP names and to initiate activity classification step.

2.6 Discussion

This section qualitatively evaluates related studies and shows which kinds of challenges have not been studied to date. Then, it brings out the advantages of our proposed approach and the difference between our main assumptions and the previously adopted ones. To this end, this section reports a set of criteria (Section 2.6.1) that qualitatively evaluate existing studies (to check if they respond to existing challenges or not). Then, it discusses these studies according to these criteria (Section 2.6.2) and outlines the main differences w.r.t our proposed approach. Quantitative comparison between related studies (in term of performance, e.g. accuracy) is not established in this section due to two main reasons:

- Quantitative evaluation resources are absent in the context of BP discovery from messaging systems : These resources are (i) A public annotated dataset related to the studied area, and (ii) Standard evaluation metrics. Up until now, only two recent studies have proposed to make these resources available in the field of BP and activity discovery from messaging systems, more precisely a public annotated dataset [8] and some evaluation metrics [84]. However, given the small size of the introduced dataset (i.e. 250) and the recent date of its publication, it has not been adopted by the proposed approaches, which means that, for evaluation purposes, the problem of the use of a public annotated dataset with large volume of data remains not resolved (which is due also to the confidentiality issue related to the logs data of emailing systems).
- The implementation of existing approaches are not publicly accessible, which means that the option of using our own dataset (annotated in the context of BP) to ensure quantitative comparison between existing studies is also not possible.

We propose to define a set of criteria according to the main challenges of the studied research area. Then, we propose to use them to qualitatively evaluate and compare related studies.

2.6.1 Evaluation Criteria

Based on the different challenges discussed in the previous chapter (see Section 1.2), an approach could be evaluated in the context of BP discovery from emails using the following criteria:
C1: Discovering multiple BP / activity perspectives; Functional, Behavioral, Data, organizational. This criterion is deduced from the research sub-questions **Q1-1** (Section 1.2.1) and **Q2-6** (Section 1.2.2).
C2: Discovering multiple activities per email. This criterion is deduced from the research sub-question **Q2-4** (Section 1.2.2)
C3: Discovering BP/ activities without disposing a knowledge a priori about them. This criterion is deduced from the research sub-question **Q2-1** (Section 1.2.2).
C4: Not requiring considerable human involvement in some key steps (This criterion is consistent with the second principle of our work, i.e. automation):

- Labeling training data;
- Selecting relevant information by manually identifying BP oriented email parts or annotating training data for recognizing them;
- Conceiving assigning rules for designating a BP element to an email;
- Setting/Tuning some user parameters (e.g. for learning algorithms);

2.6.2 Synthesis

Table 2.1: Synthesis on related studies according to the evaluation criteria C1, C2, C3 and C4

Approaches	**Criteria**									
	C1				***C2***	***C3***	***C4***			
	Functional	***Behavioral***	***Data***	***Organizational***			***Label training data***	***Select relevant data***	***Conceive assigning rules***	***Set user parameters***
Activities approaches										
[25]	x							x	x	
[50],[49]	x							x	x	x
[65]	x			x			x	x	x	
[18],[76]	x				x			x	x	x
[77],[43]	x				x			x	x	x
[86]	x		x	x		x			x	
[48]	x		x	x	x	x		x	x	
[54]	x				x		x			x
BP approaches										
[4]	x	x		x			x	x		x
[11, 51]	x	x					x		x	
[46, 45]	x	x				x	x	x	x	
[55]	x	x						x	x	
[60]	x	x						x	x	x
[24, 22]	x	x					x	x	x	
[7]	x	x						x	x	
[47]	x	x			x			x	x	
The approach that we target in this work	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	

Table 2.1 summarizes the features of the presented approaches according to previously introduced criteria C1, C2, C3 and C4.

For the first criterion (**C1**), BP discovery approaches have been mainly interested in discovering BP according to their functional and behavioral perspectives. The other perspective types (data and organizational) are almost not discussed. Only Van der et al. have studied in *EmailAnalyzer* [4] the organizational perspective by proposing two actors' involvement diagrams ; (i) task person diagram showing the relations between activities and actors, and (ii) case-person diagram showing the relations between actors and instances. Additionally, they have studied the social network of activities' related emails' users. For activity discovery approaches, some of them handled data and organizational perspectives in addition to the functional perspective [86, 48]. They discovered data perspective by detecting some predefined data types such as attached documents and email-included web pages descriptions. As for the organizational perspective, they discovered it by defining activity actors (from emails interlocutors) and their organizational (e.g. manager) and business (e.g. trading) roles. Soares et Al [86] have additionally associated to these actors weights denoting their degree of involvement. Only few studies as noticed have been focused on data and organizational perspectives. A lot of aspects were then not largely studied (e.g. actors contributions) and others were not handled to date (e.g. considering artifact notion when defining data perspective).

For the second criterion (**C2**), the question of activity multiplicity was only investigated by a few approaches discovering activities. However, it was not largely studied in the context of BP discovery where other research questions would be raised; how to identify the timestamps or the order of execution of activity corresponding events relatively to the events appearing in the same email or in other emails? (**Q2-5**, Section 1.2.2) Only Jlaitaty et al. [47] have partially handled such question by identifying the relative order of activities in the same email. However, the relative order of activities belonging to different emails was not concretely studied. In fact, it was most often assumed, for a pair of temporarily successive emails in the same BP trace, that the timestamp execution of activities of the first email precede those of the next email, which is not always the case. Let take the example of the two successive emails $email_3$ and $email_4$ in Figure 1.1 where the second email (i.e. $email_4$) is the response of the first one (i.e. $email_3$). The first email $email_3$ contains the activity 'create new deal' with a request execution purpose. We can notice from the second email that this requested activity was finally not executed and another activity was performed instead (i.e. 'extend deal'). This means that there is no precedence temporally relation between the requested activity in the first email and the executed one in the second email.

As for the third and the fourth criteria (**C3** & **C4**), existing studies still require human involvement at algorithmic level even when integrating unsupervised techniques [46, 24, 65] (**C4**). Most of them relied on integrating supervised learning techniques [18, 55, 76, 48] or manual intervention to select relevant information [86]. Actually, this needs knowing BP elements in advance (**C3**) which is not always feasible and involves human intervention (e.g. for labeling training datasets) [48]. Additionally, this limits the knowledge that could be discovered from messaging systems to the predefined one.

At algorithmic level, we can notice that existing studies make a set of unrealistic assumptions concerning the following points:

P1: The presence of BP elements in emails: Some studies supposed that BP elements are explicitly mentioned in emails (e.g. Van et Al. [4] assume that BP names are explicitly included in the email subject). This is actually limited as email senders do not necessary include such elements explicitly in emails in a regular way and with unique tags. Let take the example of Figure 1.3 reporting an email sent in the context of trading power activities. We can notice that the identifier of the BP instance is absent in the subject and the main body of this email. Meanwhile, the instance identifier is explicitly mentioned in the emails $email_1$, $email_2$, $email_3$ and $email_4$ (i.e. deal 454057) in Figure 1.1, which report activities of trading gas energy.

P2: The purpose behind using emailing systems: Most of existing studies suppose that, in the context of BP, one email is restricted to the execution purpose of one activity [4, 46, 55]. That's why they assume that only one activity event could be assigned to one email and its timestamp is similar to that of the email. Nevertheless, as previously explained (at the level of Section 1.2), one email could concern multiple activities with various purposes (e.g. activity request execution, activity execution, information about activity execution). This induces that multiple events could be assigned to one email and that their timestamps do not necessary correspond to that of the email. As it was detailed in Section 2.4.2, in the context of discovering activities from emailing systems, there have been existing studies that dealt with activity recognition by using speech act theory based methods [77, 18, 43, 76]. However, these studies have reduced BP activities into these speech acts categories which is limited; activities differ from one BP to another and speech acts would further specify the sender purpose from introducing them in emails. Consequently, activity recognition step remains required in the presence of speech act discovery step. Taking the example of the *'intention'* speech act used in Figure 1.1, it could not by itself describe in which activity context it was employed by the sender. By correlating it with *'replace deal'* activity, we could identify the purpose behind introducing this activity in the email.

P3: The outcome of the similarity of two emails: This concerns the BP elements that can be deduced from the similarity of all the textual content of two emails, especially BP and activities [25, 50, 49, 65, 46, 60]. For instance, they suppose that if the textual contents of two emails are similar then they belong to the same activity or to the same BP. Indeed, this is equivalent to suppose that the textual data expressing the concerned type of BP elements are often dominant over other data expressing other information types. Such supposition would be not often verified due to the mixed content and the granularity expression variance of BP elements within emails; emails could contain textual data that express the handled activity, instances or BP (or perhaps other topics). Additionally, the relative size of these latters varies from one email to another (for more details, see the discussed examples in Section 1.2.2 at the level of the third point). This means that we don't know the actual textual data that have induced a similarity between two emails.

P4: The lowest granularity expression level of speech acts and activities in emails: Existing approaches have considered an email [46, 24, 22, 55, 11, 51, 52, 65], an email subject [4] or an email sentence [77, 18, 43, 76, 86, 48, 52] as the lowest structural level that would express: (1) one activity, and/or (2) one sender speech act. In the context of non-controlled

textual data as emails, the expression of these two elements would not be constrained by emails' structure or punctuation. Employees could then express more than one activity instance with different speech acts even in the same sentence. Taking the example of the email $email_5$ in (Fig. 1.3); In the first sentence, the sender informs recipient(s) about one activity instance of opening a long trading position. As for the last sentence, the sender introduces three activities instances with two different speech acts; (i) cut a deal (colored by grey) and replace a deal (framed by dashed rectangle) in the form of intention , and (ii) purchase energy (framed by continuous line) in the form of information about execution.

In this work, we propose to simultaneously handle the above criteria by adopting different assumptions to be more coherent with the non-structured and the non-controlled nature of emails. The last line in Table 2.1 reports the answered criteria. The algorithmic assumptions adopted in this work concern the same points stated earlier and are as follows:

- **A1:** No prior knowledge about BP elements (**P1**, **C3**, **C4**);
- **A2:** Various use of emails in the context of BP activities (**P2**); Employees are supposed to use emails for requesting, informing, expressing intentions or requesting useful information for performing BP activities (**C1**);
- **A3:** Various granularity levels of BP elements expression (**P4**); These granularity levels are assumed to be defined according to actors' writing. Activities (including their names and their business information) are supposed to be expressed through patterns of words in emails without imposing constraints on their size (i.e. number of words). These patterns reflect the set of words that are frequently used by employees. This means that the granularity level of these BP elements are defined according to the one granted by BP actors (i.e. according to what they write). In this way, BP elements could be discovered in emails without being constrained by their structure or their punctuation, which allows the discovery of multiple BP elements, of the same or of different types, per one email (**C2**, **C1**).
- **A4:** Similarity of two emails defined according to the patterns they share (**P3**); This assumption is used for discovering the BP elements that frequently appears in emails and to associate to them their patterns of words. With this manner, the similarity between two emails induces the discovery of different BP elements with the same or with different types.
- **A5:** No sure correspondence between the timestamps of BP events and those of emails where events' related activities appear; Instead of attempting to order all events of the same BP trace in ascending chronological order (where the explicit event timestamps would be required), we rely in this work on approximating the sequencing order of each pair of successive events appearing in the same email / BP trace. Three main elements are used to ensure such approximation: (i) the verb tense (e.g. past, future) of the related activity events, (ii) the appearance order of event activities in emails, and (iii) the sequencing order of emails related events.

2.7 Conclusion

In this chapter, we presented different existing approaches that discovered BP elements from emails. We classified them according to a hierarchical categorization composed of three levels. We showed that most of them focused on discovering BP w.r.t their functional and behavioral perspectives. The data and organizational perspectives were not concretely handled to date. We additionally showed that existing challenges were not previously studied simultaneously, contrary to what we propose to study in this work. Finally, we outlined the difference between current approaches and our approach. We additionally outlined such difference in terms of the adopted assumptions. In the next chapter, we start presenting an overview on our approach integrating these assumptions.

Chapter 3

Approach Overview

Contents

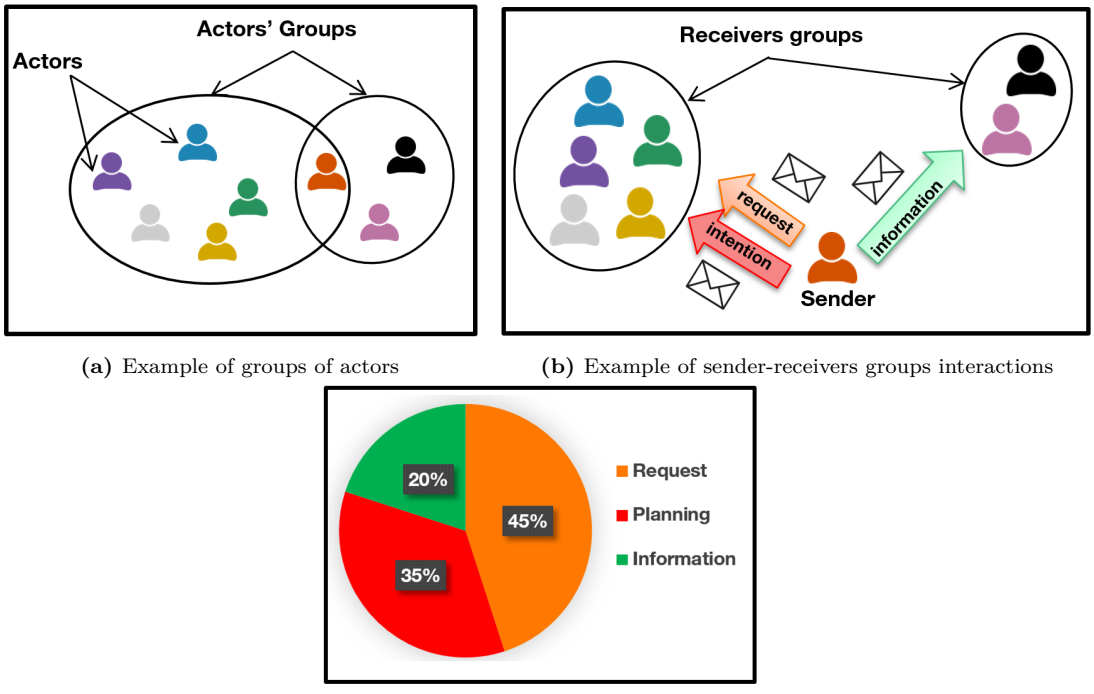
This chapter gives an overview on the proposed approach for discovering BP fragments (i.e. BP parts) from email logs data w.r.t multiple perspectives; functional, actor, data and behavioral. To this end, we give in Section 3.1 an overview on these multiple perspectives to understand what we target to discover from emails. Then, we outline in Section 3.2 the main phases of our proposed approach allowing the discovery of these perspectives.

3.1 Output overview

Due to the incompleteness of BP traces in email log data, we adopt the notion of BP fragments to be discovered from emails. A BP fragment is a set of BP elements (i.e. activities, informational entities, actors) reflecting a partial execution of one BP by a group of actors. Given the example of the BP fragment of organizing an interview, it could be described through: (i) the set of activities {'forward resume by the manager to his assistant', 'contact candidate for setting a time-slot', 'propose a time-slot' 'confirm availability'}, (ii) the set of actors {manager, assistant}, and (iii) the set of artifacts {resume, candidate}.
BP fragments are discovered in this work w.r.t these four perspectives:
`1- Functional perspective:` The functional perspective of a BP fragment reflects its composition of units of finer granularity defined by the atomic business goal that they aim to realize. These units are in our case the activities that compose the BP fragment. As for the business goal of each activity, it is reflected by its name. Taking the example of emails in the Figure 1.1, the functional perspective of the discussed BP fragment is composed of the following activities: *'flow deal', 'roll deal', 'extend deal'* and *'create deal'*.
`2- Organizational perspective:` We describe by the organizational perspective of a BP fragment the set of actors and actor groups involved in performing each activity, their related contributions and how they interact between each others in order to perform each activity.

We mean by an activity actor, an email user appearing in the interlocutor lists of activity related emails. In other words, we are interested in the actors who intervene in the emails for performing activities. We suppose that if one of them actively contributes in an activity, he/she must appear in the list of email interlocutors at some time.
We mean by an activity group of actors, the set of actors that frequently interchanging emails related to an activity (e.g. Figure 3.1 (a)). One actor could belong to more than one activity group as he/she could interchange emails with different sets of interlocutors.

We consider six contribution types of an actor to an activity: Request (i.e. request activity execution from other actors), request information (i.e. request useful information that help in activity execution), execution (i.e. perform the activity), information (i.e. inform that an activity was executed), planning (i.e. plan for activity execution) and observation (i.e. observe how an activity is performed without real intervention) (See Section 4.5.2 for more details). In the context of each activity, we associate to each actor the set of contributions that are most often made by him/her. We describe each one of them by a coefficient reflecting its ratio from his/her overall contributions made in the context of different executions of the same activity. In this way, we obtain for each activity the distribution of contributions per actor. Taking the example of Figure 3.1 (c), it describes the distribution of the contributions of one actor w.r.t one activity through the following ratios (expressed in percentage): 20% (information), 35% (planning) and 45% (request).



(a) Example of groups of actors

(b) Example of sender-receivers groups interactions

(c) Example of contributions' distribution of an actor

Figure 3.1: Illustrative example of an activity organizational perspective

To outline how actors interact between each others in order to perform each activity, we

introduce the notion of Sender-Receivers groups. A Sender-Receivers group is composed of: (i) one sender that sends activity related emails, (ii) a group of actors that frequently co-exist in the receivers list of these emails, and (iii) sender-receivers relations that correspond to the sender speech acts referring to his/her purposes when talking about the activity in these emails. We consider four speech acts in our work: information act, request act, request information act and intention act (See Section 4.5.2 for more details). Figure 3.1 (b) illustrates an example of two Sender-Receivers groups sharing the same sender. The sender sends information of activity execution for one receiver group and sends request and intention of execution for the other group.

`3- Data perspective:` This perspective describes the set of artifacts manipulated by each BP fragment. An artifact is an informational entity defined by its name (e.g. 'Gas Deal' in Figure 3.2) and a set of attributes (e.g. ID, counterparty, volume, price). We illustrate through this perspective two types of relations:

- Artifact-Activity relations to show what artifacts are manipulated by a given activity. For instance, *'create deal'* activity manipulates 'Gas Meter' and *'Gas Deal'* artifacts (see dashed lines in the FIGURE 3.2)
- Artifact-Artifact relations to show which kind of cardinalities (e.g. one-to-one, one-to-multiple) relates two given artifacts. For instance, a gas meter could be associated to multiple trading deals, while a gas deal could be associated to one gas meter for delivery purposes (see continued line in the FIGURE 3.2).

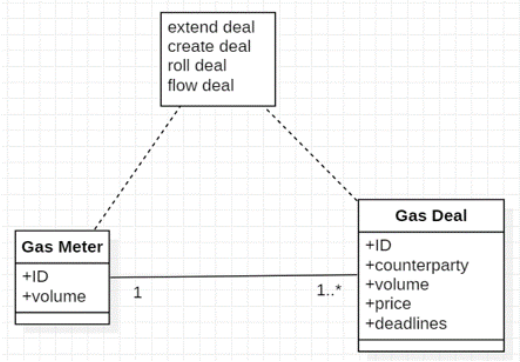
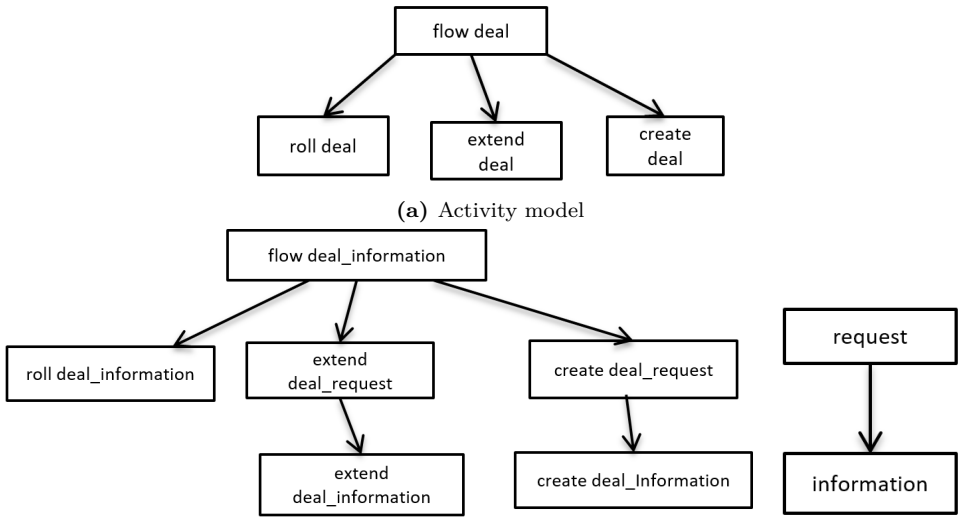


Figure 3.2: Illustrative example of a BP fragment data perspective

`4- Behavioral perspective:` Behavioral perspective of a BP fragment describes the conditions for the occurrence of event types. We mean by an event type in our work: (i) an activity, (ii) a speech act referring to the purpose when talking about an activity in an email, or (iii) an activity concatenated with a specific speech act. These conditions are described through a set of constraints forming three model types:

- Activity model illustrating sequencing constraints between activity event types. Taking the example of FIGURE 3.3 (a), it represents a graphical representation of activity



(a) Activity model

(b) Activity & SA Model

(c) SA model of 'extend deal' activity

Figure 3.3: Illustrative example of a BP fragment behavioral perspective

sequencing constraints that could be obtained from threads similar to Figure 1.1 (each edge linking to rectangular nodes reflects a sequencing constraint). They show that following a flow deal event, a gas trader could roll, extend or create a new deal;

- SA model (i.e. Speech Act model) illustrating the sequencing constraints between event types referring to speech acts (e.g. request, information, intention) of the same activity. Such model describes how emails are used when performing an activity (e.g. Figure 3.3 (c) shows how the information about extend deal execution could be preceded by a request as the case of $email_3$ and $email_4$ of Figure 1.1);

- Activity & SA model (i.e. Activity & Speech Act model) illustrating the sequencing constraints between event types referring to activities concatenated with their speech acts (e.g. 'extend deal_request' is an event type that uses the character '_' to concatenate the activity 'extend deal' and the speech act 'request'. It refers to requesting the extension of a deal in an email). Figure 3.3 (b) shows a graphical representation of sequencing constraints of activities concatenated with their speech acts, which could be obtained from threads similar to Figure 1.1;

3.2 Approach Overview

To discover BP fragments from emails, the main challenge is how to transform the unstructured log data of emails into a structured event log compatible with multiple BP perspectives discovery. To this end, we propose, as illustrated in Figure 3.4 an approach composed of three main phases: (i) Email logs pre-processing (Chapter 5), (ii) Event log generation (Chapter 5 & Chapter 6) and (iii) Event log mining (Chapter 7).

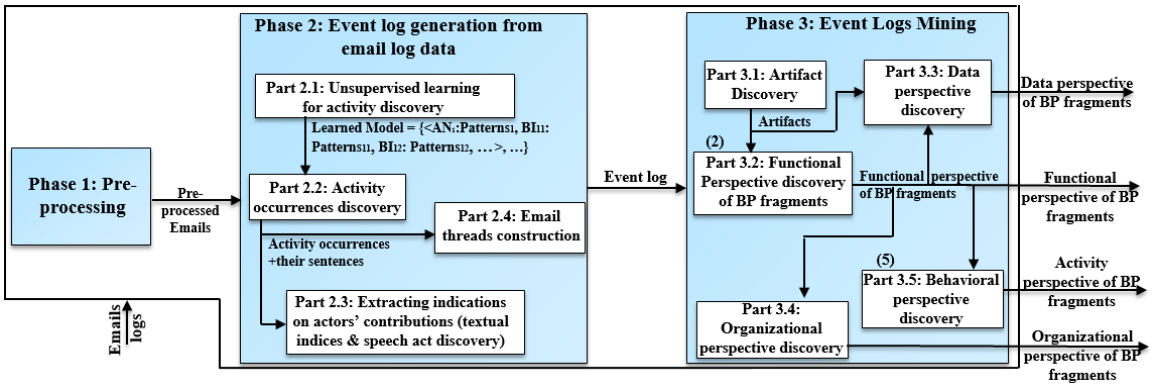


Figure 3.4: Proposed approach for multi-perspective BP fragments discovery

3.2.1 Phase 1: Email log pre-processing

At the input of this phase, each email textual content is composed of three parts: (i) subject, (ii) main body referring to the message written by the sender of the email, and (iii) conversation history referring to the trace of previous emails for which the main body was sent as a reply or a forward. This phase pre-processes the textual contents of email main bodies and subjects. It applies a set of natural language pre-processing operations (e.g. lemmatization, stopwords removing, named entities detection, etc). The goal is to return to each email a list of tuples mapping each word to some local features, e.g. raw format (e.g. *'rolled'*), position of appearance in email, lemmatized format (e.g. *'roll'*), a tag indicating if the word is a verb ('v') or not ('n'), etc.

3.2.2 Phase 2: Event log generation

This phase generates a structured event log from the pre-processed email log. Each event corresponds mainly to the occurrence (i.e. appearance) of one activity in an email and records useful BP elements enabling multiple BP perspectives discovery. These elements include:

- The occurred activity name to be able to identify BP fragments functional and behavioral perspectives;
- The occurred business information that could give indications about the name or/and the attributes of the manipulated artifacts (i.e. useful for BP fragment data perspective discovery);
- The speech act of the occurred activity referring to the sender purpose (i.e. information, request, request information, intention) of talking about it in the email. This BP element is useful for discovering BP organizational perspective and behavioral perspective;
- A tag approximating the notion of trace identifier enabling the grouping of events of the same BP trace, which is required for mining BP from any structured event log;

For generating such structured event log, activities present preliminary entities that must be first recognized. This is required to be able to discover their occurrences in emails while capturing their related useful information (e.g. business information, speech acts). To this end, this phase performs a set of algorithmic steps divided into the following parts:
`Part 2.1: Unsupervised learning for activity discovery:` This part analyses the main bodies of emails in order to discover activities. The discovered activity has three components: (i) activity name that reflects its main goal (e.g. create deal), (ii) Business data that reflect its consumed and produced data (e.g. deal identifier) and (iii) Business context information useful for understanding the business context of performing an activity (e.g. 'gas deal' implies that the deal is performed in the context of trading gas energy). This part is ensured in an unsupervised way and without disposing any a priori knowledge about the existing activities. We introduce to this end an approach based on frequent patterns of words discovery to capture common substructures (in terms of combination of words) shared by activity expressions in emails. This is ensured while tolerating using expressions having words that could be:

(i) Different but sharing the same meaning (e.g. the expressions *'change deal'* and *'convert deal'* refer to the same activity while using the synonymous words: *'change'* and *'convert'*) or the same business context (e.g. *'volume'* and *'gas'*);

(ii) Not appearing continuously or/and in the same sequential order in emails (e.g. the positions of appearance of the words *'extend'* and *'deal'* are switched in these expressions *'the deal was extended'* and *'I extended the gas deal'*. Additionally, the two words are nearly successive in the first expression while separated by the word *'gas'* in the second expression);

The ultimate goal of this part is to obtain a learned model that associates for each activity, its components and the related patterns of words. To this end, it consists of two main steps: The first one analyzes per employee emails to reduce expression variance. It aims to capture the set of frequent patterns related to the activity components (i.e. name, business data and business context information) present in emails. The second step regroups similar activities of different employees. To regroup patterns belonging to the same activity (first step) and to regroup similar activities of different employees (second step), two similarity measures are used on the basis of: (ii) words' synonyms and (ii) activity/pattern business context (defined by their related business data and business context information).
`Part 2.2: Activity occurrence discovery:` This part discovers all the occurrences of the previously discovered activities from emails that are not necessary considered in the learning part (i.e. Part 2.1). For this purpose, it uses the obtained learned model which associates to each activity component the set of patterns frequently used to express it. This allows the discovery of the occurrences of multiple BP elements per one email without being constrained by email structure or punctuation. Each discovered activity occurrence is composed of three elements referring to the occurrences of its component. Each activity component occurrence is characterized by the position of appearance of its related pattern's words and their corresponding raw format. Given the activity name *'roll deal'*, its occurrence in the email $email_2$ in Figure 1.1 is characterized by the tuple of raw words *(*'rolled', 'deal') and the tuple of

integers (7, 4) that captures the position of their appearances. More examples concerning activity component occurrences are shown in Figure 3.5.
`Part 2.3: Extracting indications on actors contributions`: This part performs two main steps. In the first step, it extracts textual indices that would refer to the real performers of activity occurrences (e.g. personnel pronoun). In the second step, it discovers the speech acts related to each activity occurrence to be used (in addition to actors' indices) to deduce interlocutors contributions. For instance, for the activity 'roll deal' occurring in the email $email_2$ in Figure 1.1, the corresponding speech act is an information about execution. In this step, we propose two methods for discovering speech acts for an occurring activity:

- Rules based method: It is based on defining a set of rules, summarized in the form of a decision tree, which maps speech acts to activity verb grammatical features (e.g. tense, raw format);
- Supervised method: It applies any existing supervised learning algorithm. It does not require a human expertise to define prediction rules. However, it leverages a labeled training dataset;

`Part 2.4: Email threads construction`: A thread in our work approximates the notion of trace in process mining. It is composed of a set of conversations having relevant information values (e.g. business data values, email addresses) in common. A conversation is composed of a set of emails having reply or forward relations (for instance, in Figure 1.1, [$email_1$, $email_2$] and [$email_3$, $email_4$] form two different conversations respectively). They are obtained by matching emails in terms of their textual contents and their conversation histories.
The relevant information values that regroup conversations into threads approximate BP instance identifiers. An email could belong to more than one thread as it could introduce more than one instance identifier. Figure 1.1 illustrates a real example of a thread of emails retrieved from Enron dataset. The thread is composed of four emails belonging to two conversations. It reports interactions between employees in the context of a trading gas instance. This instance is identified by the relevant information value *'454057'* of the associated deal number;

To better understand the structure of the generated event log from Phase 2 (Figure 3.4), we show in Figure 3.5 an example of an event log extract. This extract could be obtained from an email log including $email_1$ and $email_2$ shown in Figure 1.1 and the email shown in Figure 1.3. For each event of identifier Ev_ID, the event log extract reports the following attributes:

- The occurred activity $Act_o = (AN_{occ}, BD_{occ}, BC_{occ})$ where AN_{occ}, BD_{occ} and BC_{occ} refer respectively to the occurrences of its components: activity name, business data and business context;
- The speech act of the occurred activity (SA);
- Textual indices concerning the performers of the occurred activity (At_{ind});
- The related relevant information values (I_{values});

- Email attributes where the activity occurred (em): ID to refer to its unique identifier, *timestamp* to refer to its sending time, *Sender* and *Recipients* to capture email interlocutors and $ConvID$ to capture the conversation ID to which the email belongs;
- The set of threads ID to which the event belong (Th_{id});

Event	Act_o			SA	At_{Ind}	I_{values}	em					Th_{id}
Ev_ID	AN_{occ}	BD_{occ}	BC_{occ}				ID	timestamp	Sender	Recipients	ConvIDs	
e_1	('flow deal', ['flowed', 'deal'], [7,4])	{('deal numeric' ,['deal','412219'], [4,5]), ('meter numeric',['meter','1601'], [2,3]), ('deal numeric' ,['deal','45057'], [11,12]), ('meter numeric',['meter','5192'], [9,10])}	{('meter deal', ['meter', deal'], [2,4])}	'information'	[]	['deal numeric_deal 412219'; 'meter numeric_meter 1601', 'deal numeric_deal 45057', 'meter numeric_meter5192']	'<1552589.1075853972210.JavaMail.evans@thyme>'	'Fri, 10 Nov 2000 06:17:00 -0800 (PST)'	'aimee.lannou@enron.com'	['daren.farmer@enron.com']	'C1'	['Th_1']
e_2	('flow deal', ['flowed', 'deal'], [14,11])		{}	'information'	[]							
e_3	('roll deal', ['rolled', 'deal'],[2,3])	{('deal numeric' ,['deal','45057'], [3,4]), ('meter numeric',['mtr','5192'], [7,8])}	{}	'information'	[]	['deal numeric_deal 45057', 'meter numeric_meter5192']	'<29717536.1075855038538.JavaMail.evans@thyme>'	'Wed, 15 Nov 2000 02:24:00 -0800 (PST)'	'daren.farmer@enron.com'	['aimee.lannou@enron.com']		
e_4	('cover flow', ['cover','flow'],[3,4])			'intention'	[]							
e_5	('open short position', ['short', '100', 'mws'], [1, 2, 3])	{('numeric mw' ,['numeric','100'], [2,3]), ('hour0end numeric numeric',['hour ending','7', '22'], [5,6]), ('orgname', ['Ercot Asset'], [4]), ('orgname', ['El Paso'],[20]), ('orgname', ['Empower'], [26]), ('numeric mw' ,['numeric','100'], [28,29]), ('orgname', ['Ercot'], [30]), ('orgname', ['bal-day'], [38])}	{}	'information'	['we']	[]	'<2353515.1075855236920.JavaMail.evans@thyme>'	'Wed, 26 Sep 2001 10:19:15 -0700 (PDT)'	'm..forney@enron.com'	['joe.capasso@enron.com', 'l..day@enron.com', 'joe.errigo@enron.com', 'alexander.mcelreath@enron.com', 'jeffrey.miller@enron.com', 'steve.olinde@enron.com', 'eric.saibi@enron.com']	'C3'	['Th_3']
e_6	('display deal', ['shows', '100', 'mws'], [27, 28, 29]}			'information'	['enpower']							
e_7	('cut deal', ['cut', 'deal'], [33, 31])			'intention'	[]							
e_8	('replace deal', ['replace', 'deal'], [34, 31])			'intention'	[]							
e_9	('purchase power', ['power', 'purchased'], [36, 35])			'information'	[]							

Figure 3.5: Example of event log extract

3.2.3 Phase 3: Event log mining

This phase analyses the obtained event logs to discover BP fragments w.r.t their functional, data, organizational and behavioral perspectives. At this level, two elements remains missed: (i) artifacts representing the informational entities in the data perspective of BP fragments, and (ii) the organization of activities into BP fragments (i.e. functional perspective), which enables projecting the obtained event log on BP fragments to discover their data, organizational and behavioral perspectives. To this end, this phase performs the following parts:

`Part 3.1: Artifact discovery:` We discover artifacts by applying an overlapping clustering[1] on activities and their business information (i.e. business data and business context information) as one activity could manipulate multiple artifacts (e.g. 'extend deal' in Figure 3.3 manipulates two artifacts: 'Gas Deal' and 'Gas Meter');

`Part 3.2: Functional perspective discovery of BP fragments`: This part applies an overlapping clustering on BP elements (i.e. activities, artifacts and actors) to organize them into BP fragments, so that one BP element could belong to multiple BP fragments. To this

[1] An overlapping clustering allows data to belong to multiple groups

end, the similarity between each pair of BP elements (which is required for ensuring clustering) is inferred from their degree of coexistence in the same BP threads.

Part 3.3: Data perspective discovery: This part discovers the BP fragments' data perspective (e.g. Figure 3.2). At this level, artifacts are discovered and mapped to the activities that manipulates them (through Part 3.1), which allows us to identify Artifact-Activity relations. Additionally, artifacts are associated to each BP fragment (through Part 3.2). Artifact-Artifact relations remains to be discovered. To this end, this part projects the obtained event log on BP fragments. Then, it uses the coexistence coefficients between each pair of artifacts in terms of appearance in the same threads to identify those that are in relation (i.e. those that highly coexist) and to estimate their cardinalities.

Part 3.4: Organizational perspective discovery: This part projects the obtained event log on the activities of a given BP fragment to obtain their related sublog. Then it analysis these sublogs to discover the organizational perspective of each BP fragment activity by mining the following elements:

- The set of actors' groups (e.g. Figure 3.1 (a)) involved in performing the activity are inferred from the set of actors that highly coexist in the same interlocutors lists of activity related emails;
- The Sender-Receivers groups (e.g. Figure 3.1 (b)) of an activity are inferred from actors of the same group that highly coexist in the same receivers lists of emails sent by the same sender;
- Actors contributions towards performing the activity (e.g. Figure 3.1 (c)). These contributions are inferred from speech acts of activity occurrences in emails through: (i) a direct mapping between speech acts and some contributions types (e.g. the speech act intention refers to a planning contribution), or (ii) considering additional actors' indices retrieved from activity related emails (e.g. personnel pronouns like 'I' and 'you' could refer to the real executor of the activity) or by tracking actors reactions towards received emails (e.g. actors answering to execution requests would be the real activity executors);

Part 3.5: Behavioral perspective discovery: This part analyzes the sublog of each BP fragment to discover three model types; (i) Activity model (e.g. Figure 3.3 (a)), (ii) SA model (e.g. Figure 3.3 (c)) and (iii) Activity & SA model (e.g. Figure 3.3 (b)). To this end, this part generates sequencing constraint candidates between event types from events appearing in emails and threads. It uses their positions of appearance in emails/threads and their related speech acts to estimate their relative execution order. Therefore, it filters these constraints candidates on the basis of their number of occurrence and their confidence measure.

Chapter 4

Approach Formalization: Meta-Model & Main Notations

Contents

4.1 Introduction

This chapter describes the meta-model of our proposed approach and formalizes the definitions of the main entities appearing in each phase.

The first main requirement that needs to be fulfilled by our metamodel is to allow a multi-perspective modeling of BP fragments that takes into account the activities, the produced/consumed data and the involved actors. To this end, we propose a metamodel that follows an artifact-centric approach [2] which does not only focus on the activities, but also

treats the data as first-class citizen. We also adopt a declarative approach [75] to describe the behavioral perspective models of our discovered BP fragments since declarative languages provide more flexibility for modeling BP with high variability (which is the case of employee-driven processes executed through emails).

The second main requirement that needs to be fulfilled is to take into consideration emailing systems specificities regarding: (i) their operational mode, and (ii) their use in the context of BP. To this end, we integrate in our metamodel these four parts (as depicted in Figure 4.1):

1. Input model (colored in blue) to formalize the email log data disposed at the entry of our approach. We describe in this way the execution trace of emailing systems according to their own operation mode;
2. Intermediate output model (colored in purple) to illustrate the structure of the activities to be discovered at the level of Part 2.1 and required for generating a structured event log in Phase 2 (see Figure 3.4);
3. Event log model (colored in green and gray) to illustrate the main entities appearing at the level of the event log structured format to which the input (i.e. email log) is transformed (at the level of Phase 2). We formalize such format w.r.t to the additional or the available BP elements that could be retrieved from emails, compared to the most common event logs. We model at this level the main entities reflecting how emails are used in the context of BP, to show the link between email log and our event log structure.
4. Final Output model to formalize the main entities (colored in orange) appearing in the final results generated by Phase 3 in our approach (see Figure 3.4);

To validate our approach metamodel, we present a real life case study conducted on real emails collected from mailboxes of Orange and Enron employees. We show the results of two experiments carried out to prove the validity of the main design choices adopted in this metamodel. Some parts in this chapter were published in the International Conference on Services Computing (SCC) [28].

The remainder of this chapter is structured as follows. We explain our proposed metamodel according to its four parts: input model in Section 4.2, intermediate output model in Section 4.3, event log model in Section 4.4 and the final output model in Section 4.5 . In Section 4.6, we talk about the validation of our approach metamodel. Finally, we conclude in Section 4.7.

4.2 Input: Emails Log

In what follows, let $\mathcal{ID}$ be the set of email identifiers and let $\mathcal{AD}$ be the set of emails addresses. We formalize the email log (Input Model in Figure 4.1) disposed at the entry of our approach as follows:

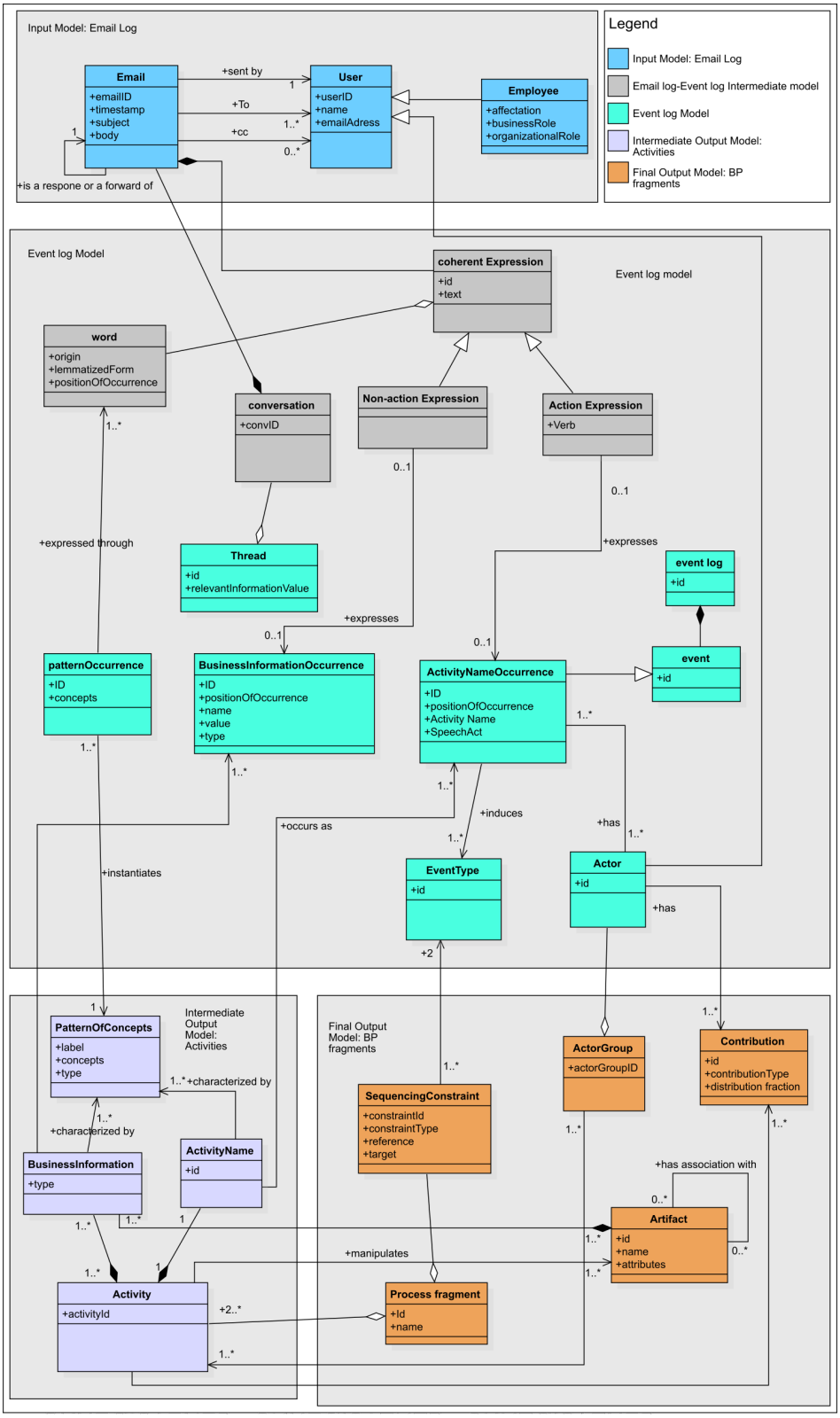


Figure 4.1: Approach Meta-Model

Definition 4.1. ***(Emails Log Data: EmL)*** *An emails log data (EmL) is a set of emails, each one has the following format: email* $= (emailID, timestamp, sender, receivers, subject, body)$ *where:*
- $emailID \in \mathcal{ID}$ *is the email unique identifier;*
- timestamp is the email timestamp;
- $sender \in \mathcal{AD}$ *is the email sender (represented in our meta-model through the entity 'User' and the association '+sent by');*
- $receivers \in \mathcal{AD}^*$ *is the list of email receivers with different status: to, cc (represented in our meta-model through the entity 'User' and the associations '+to', '+cc');*
- subject is the email subject;
- $body = (mainbody, convhist)$ *is the email body composed of a main body (mainbody) and a conversational history (convhist);*

Each email user (sender/receiver) can be internal (i.e. employee) or external (e.g. client) to an organization (e.g. company). In the case of an internal, he/she is generally characterized by two roles' types most often provided by a company: (i) Organizational role that defines his/her position in a company organizational hierarchy (e.g. manager, assistant), and (ii) Business role that defines the business nature of tasks he/she performs, e.g. trader, deliverer. For externals' roles, they are denoted "external" in this report. In what follows, let $\mathcal{OR}$ and $\mathcal{BR}$ denote respectively the set of organizational roles and the set of business roles.

4.3 Intermediate Output Model: Frequent Activities

Frequent activities are those that are repetitively expressed in emails and are likely to belong to a BP. These activities are discovered in Step 2.1 (Section 3.2.2) in order to generate our structured event log. An activity (see the entities colored in purple in Figure 4.1) is formally defined as follows (Definition 4.2):

Definition 4.2. ***(Activity)*** *An activity is defined as* $Act = (AN, BD, BC)$ *such that:*
- AN *is the name of the activity that reflects its main goal;*
- $BD = \{bd_1, ..bd_p\}$ *is a set of business data used and generated during activity execution of size* $p \in \mathbb{N}$*;*
- $BC = \{bc_1, ..bc_d\}$ *is a set of business context information (of size* $d \in \mathbb{N}$*) referring to the set of words or combination of words useful for understanding the business context of an activity execution;*
- $BI = BD \cup BC$ *is the overall set of business information (BI) of the activity;*

An activity is defined as a composition of three components (activity name, business data and business context information). Taking the example of the activity creating trading deals; (1) the activity name (AN) is *'create deal'*, (2) the set of {*'deal price'*, *'deal identifier'*} are included in its business data (BD), and (3) *'power deal'* referring to a trading deal applied on electricity power energy belong to its business context information (BC).

In what follows, $\mathcal{A}$ will denote the set of all activities. $\mathcal{AN}$, $\mathcal{BD}$ and $\mathcal{BC}$ will denote the sets of activity names, business data and business context information respectively. Finally, $\mathcal{BI}$ will denote the set of $\mathcal{BD} \cup \mathcal{BC}$.

Each activity component is discovered in our work in the form of a set of *patterns of concepts* (Definition 4.4) used by employees to express it. We introduce the notion of *concept* to regroup a set of synonyms supporting one potential meaning and having the same context of use in natural language (e.g. the set of verbs {*'purchase'*, *'buy'*} forms one concept). A pattern of concepts reflects a set of frequent combination of words that are frequently used by employees to express an activity component in their emails. As for its size (i.e. number of concepts per each pattern), it depends on the size of the frequent combination of words: (i) to which it refers, and (ii) that are used by the employees to express the concerned activity component.

In what follows we formally define the notion of *concept* and *pattern of concepts*. To this end, we adopt the following notations:

- Let $\mathcal{W}$ be the set of lemmatized words (i.e. words in basic format) ;
- Let $LEM : EmL \rightarrow \mathcal{W}^*$ be the function that returns for each email, the list of lemmatized words of the raw words appearing in its main body (after removing stopwords). Supposing that the main body of one email is composed of the following string: *"Due to time constraints, I have not researched pricing and volumes."* After splitting it, lemmatizing its raw words and removing stopwords, we obtain the following list: [*'time', 'constraint', 'research', 'pricing','volume'*]
- Let $\mathcal{T}$ be the set of part of speech categories that refer to the grammatical properties of words (e.g. noun, verb) without referring to their tense (e.g. past, present) or to their number (e.g. plural);
- Let $POSTag : EmL \times \mathcal{W} \times \mathbb{N} \rightarrow \mathcal{T}$ be a part of speech function that returns for each word $w \in \mathcal{W}$, the corresponding part of speech category tag $t \in T$ when it appears in an email $e \in EmL$ at a position $p \in \mathbb{N}$;
- Let SynFc: $\mathcal{W} \times T \rightarrow \mathcal{W}^*$ be a synonymy function that returns the synonyms of a word $w \in \mathcal{W}$ in respect to a part of speech category $t \in T$;

Definition 4.3. ***(Concept)*** *A concept is a tuple $C = (L_w, t)$ composed of a list of lemmatized words $L_w \subset \mathcal{W}$ and a part of speech category tag $t \in T$ where $\forall (w_i, w_j) \in L_w \times L_w$:*

- $\exists\ (em_i, p_i) \in EmL \times \mathbb{N}$ *and* $\exists\ (em_j, p_j) \in EmL \times \mathbb{N}$ *such that* $LEM(em_i)[p_i] = w_i$ *and* $LEM(em_j)[p_j] = w_j$
- $t = POSTag(w_i, em_i, p_i) = POSTag(w_j, em_j, p_j)$*;*
- $w_i \in SynFc(w_j, t)$*; AND*
- $w_j \in SynFc(w_i, t)$*;*

A concept regroups a set of synonyms supporting one potential meaning and having the same part of speech category (e.g. the set of nouns {*'deal'*, *'trade'*} forms the concept C_{ex} =({*'deal'*, *'trade'*}), *'noun'*). If one concept appears in an email at a defined position, it will be expressed using a single word belonging to its synonyms.
Each words' pair belonging to a concept must verify these two criteria: (i) similarity in terms of part of speech category, and (ii) reciprocal relations of synonymy, which means that each word must belong to the synonyms of the other word. We require these two criteria so that one concept supports the expression of a single meaning. We take the following examples to show how the meaning of one concept could diverge if these two criteria are not verified:

- For the part of speech criterion: We take the example of the word 'deal'. This word has a synonymy relation with the word 'trade' if it is of 'noun' part of speech category. Nevertheless, it has a different meaning w.r.t 'verb' part of speech category. For instance, 'to deal' would have the meaning of 'to cope with', which is different from the meaning of 'to trade'.
- For the synonymy reciprocity criterion for all words' pairs forming a concept: We require it as the synonymy relation is not necessary a transitive relation. This means that, for a synonymy relation $\mathcal{R}$, if we dispose three words w_1, w_2 and w_3 such that $(w_1\mathcal{R}w_2 \bigwedge w_2\mathcal{R}w_3)$, it is not sure that we will obtain $w_1\mathcal{R}w_3$. This would lead to two potential meanings in the same set of words formed by w_1, w_2 and w_3. We take the example of the words' pairs ('set', 'arrange') and ('set', 'define'). For each pair, words are reciprocally synonyms, however, 'arrange' and 'define' are not reciprocally synonyms. If the words 'define', 'set' and 'arrange' belong to the same concept, this latter would express two potential meanings referring to two different activities (organizing or defining something).

In what follows, let $\mathcal{C}$ denote the set of concepts.

Definition 4.4. ***(Pattern of Concepts)*** *A pattern of concept PC is a set of concepts of size* $n_P \in \mathbb{N}\backslash\{0,1\}$ *where;*

- $\forall j \in [1, n_P]$, $\exists\ C_j \in \mathcal{C}$; *AND*
- $PC = \{C_1, .., C_{n_P}\}$;

Let take the example of PC_1={({*'purchase'*, *'buy'*}, *'verb'*), ({*'power'*}, *'noun'*)}, it is one of the patterns of concepts that could be associated to the activity name AN_1 =*'purchase electricity power'* appearing in the email of Figure 1.3.

A set of patterns of concepts associated to the same activity component are patterns that have some concepts sharing the same business context of use (which means without necessary having synonymous relations). Taking the example of PC_2={({*'buy'*, *'purchase'*}, *'verb'*), ({*'numeric'*},*'noun'*), ({*'mw'*},*'noun'*)}, it is associated to the same activity name AN_1. This allows the detection of AN_1 in an email by only referring the purchased quantity

(*'numeric'*) of power in megawatt (*'mw'*) rather than explicitly mentioning the word 'power' or another word belonging to its synonyms.

Patterns of concepts associated to the same activity differ from one another depending on the activity component to which they refer. We differentiate between them based on the following rules that we equally use in the next chapter (i.e. Chapter 5) when discovering activities:

- Each activity name pattern must contain at least one verb to reflect the action made in the context of the related activity;
- Each business data pattern does not contain verbs and must contain tags referring to numeric values (e.g. numeric, pricenumeric, datenumeric) or to named entities' types (e.g. location, person name). A business data pattern actually reflects a business data name. The numeric or named entity tag that it contains specifies the business data type (e.g. person name tag indicates that it is of string type, datenumeric tag indicates that it is of date type, etc). When occurring in emails, business data appears in the form of business data values. This means that the tag contained in business data patterns appears in the form of values having the same tag type (e.g. the business data pattern 'deal numeric' appears in the email $email_2$ of Figure 1.1 in the form of the value 'deal 454057');
- Business context patterns must not contain verbs or tags referring to numeric values or named entities;

Such relations between activities and patterns is illustrated in our meta-model through the association *'+characterized by'* between the entities: (i) *'PatternOfConcepts'*, and (ii) *'ActivityName'* and *'BusinessInformation'* (i.e. referring to business data and business context information). We give additional examples in Table 4.1 concerning such relations. For each

Table 4.1: Examples of patterns of concepts characterizing activity components

id	**Activity component**	**Type**	**Patterns of concepts**
1	organize interview	Activity name	P_{11} = {({*'arrange'*, *'organize'*}, *'verb'*), ({*'interview'*}, *'noun'*)}
2	interview duration	Business data	P_{21} = {({*'numeric'*, *'noun'*), ({*'hour'*}, *'noun'*)}
3	phone interview	Business context	P_{31} = {({*'phone'*, *'telephone'*}, *'noun'*), ({*'interview'*}, *'noun'*)}
4	see file	Activity name	P_{41} = {({*'see'*, *'check'*, *'verify'* }, *'verb'*),({ *'file'*, *'document'*}, *'noun'*) }
5	change deal	Activity name	P_{51} = {({*'change'*, *'modify'*}, *'verb'*), ({*'deal'*}, *'noun'*) }
6	number of deal	Business data	P_{61} = {({*'deal'*}, *'noun'*), ({*'numeric'*}, *'noun'*) }
7	purchase electricity power	Activity name	P_{71} = {({*'purchase'*, *'buy'* }, *'verb'*), ({*'power'*}, *'noun'*) }; P_{72} = {({*'purchase'*, *'buy'* }, *'verb'*), ({*'numeric'*}, *'noun'*), ({*'mw'*}, *'noun'*)}
8	power quantity	Business data	P_{81} = {({*'numeric'*}, *'noun'*), ({*'mw'*}, *'noun'*)}

label of activity component (e.g. *'phone interview'*), we provide its type (i.e. activity name,

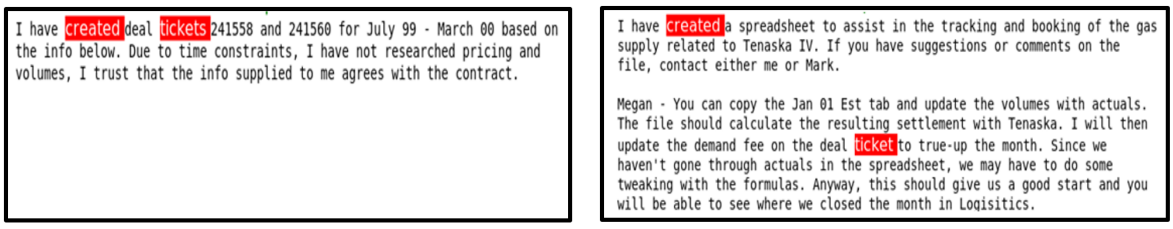
business data or business context information) and example(s) of pattern(s) of concepts characterizing their expressions in emails.

4.4 Event Log Model

Our structured event log (illustrated through the green entities in our meta-model) is generated from Phase 2 in our approach. It is used to discover BP fragments including their functional, data, organizational and behavioral perspectives. Each event in our event logs corresponds to the occurrence of an activity in an email (inheritance association between the entities *'Event'* and *'ActivityNameOccurrence'*). To enable the discovery of BP fragments w.r.t multiple perspectives, we have defined event log's attributes analogous to those adopted in the most common event logs, in the way that they are consistent with emails specificities. Additionally, we have enriched them with a new one referring to activity speech act. In what follows, we explain, in Section 4.4.1, the relation between emails and activity occurrences where we outline our main assumptions describing how activity occurrences are expressed in emails. Then, in Section 4.4.2, we formalize the structure of our event log.

4.4.1 Activity occurrences' expression in emails

We suppose that activity components (i.e. activity names, business data and business context information) occurrences are expressed in emails through coherent expressions. We define a coherent expression as a low dispersed set of words that contribute in precising the same idea (e.g; 'create ticket') in the email. Taking the example of two emails' extracts (A & B) retreived from enron dataset and illustrated in Figure 4.2; A and B share the set of words {'create', 'ticket'} which is low dispersed in A and highly dispersed in B. Both words in A contribute to the expression of the idea 'ticket's creation', while in B, each word contributes to the expression of a different idea ('spreadsheet creation' and 'ticket update'). This shows that low dispersity of words, as in A, is essential to express activity components. Coherent expressions are of two types:



(a) Email extract A **(b)** Email extract B

Figure 4.2: Email extracts retrieved from Enron database

1. Action expression containing at least one verb and that would be used to express an activity name. This is shown in our metamodel through the association *'+expresses'*

that links the entity *'Action expression'* (that inherits the entity *'coherentExpression'*)) to the entity *'ActivityNameOccurrence'*;

2. Non action expression that does not contain verb and that would be used to express business data or business context information. This is shown in our metamodel through the association *'+expresses'* that links the entity *'Non-Action expression'* (that inherits the entity *'coherentExpression'*) to the entity *'BusinessInformationOccurrence'*

4.4.2 Event log structure

Let $\mathcal{O}$ denote the set of activity components (activity names, business data or business context information) occurrences in emails. These BP elements are recorded by each event in our event log. Formally, we define our event log structure as follows;

Definition 4.5. ***(Event Log)*** *An event log (EL) is a set of events* $ev = (id, Act_o, SA, em, I_{values}, Th_{id})$ *where:*

- $Act_o = (AN_{occ}, BD_{occ}, BC_{occ}) \in \mathcal{O} \times \mathcal{O}^* \times \mathcal{O}^*$ *is the occurring activity;*
- $em = (ID, timestamp, ConvIDs, Sender, Recipients)$ *is the email in which the activity occurred. It is a tuple of values of email attributes denoting respectively its ID , its timestamp, IDs of conversations to which it belongs, its sender and its recipients with different status (To, Cc,..);*
- SA *is the speech act related to the occurring activity name;*
- At_{Ind} *is the set of textual indices that refer to the performer of the occurring activity;*
- I_{values} *is the set of relevant information values while* $I_{values} \subset BD_{occ} \cup Recipients$*;*
- Th_{id} *is the set of thread ids to which em belongs;*

Compared to the most adopted event logs, our event log structure differs in four ways:
- The notion of ***activity occurrence*** is introduced as an analogous to the well-known concept of ***activity instance*** in process mining. The main difference is that activity occurrence refers to the appearance of the name of the activity (and optionally of its business information) in an email without: (i) necessarily being executed, or (ii) disposing precise information concerning its instantiation. This is consistent with the specifities of emailing systems that: (i) do not require employees to compose all the attributes of an activity instance in an email, and (ii) could be used for different purposes in the context of BP activities, which are not limited to the execution purpose.
- The notion of ***relevant information value*** is introduced to approximate the notion of ***instance ID***. In fact, instance ID are not necessary included in emails when referring to activities. In the case of indicating them, they would not be expressed in an explicit way. Relevant information values are selected after instantiating receivers and business data related to the occurrences of each activity. They are selected in the way that they tend to uniquely

characterize each occurrence of each activity. In our case, they refer to receivers of business data values having largely distinct instantiating values.
- The notion of ***thread*** is also introduced as an analogous to the concept of ***trace*** in process mining. The difference is that it refers to a set of emails conversations without guaranteeing that emails do not mention more than one instance (This is due to the non-controlled nature of emails where employees could introduce multiple instances in the same email, e.g. they could request the execution of some operations on two instances in the same email).
- The notion of ***actor indices*** is defined to capture the set of textual terms that could refer to the real performers of the occurring activities. For instance, the sender of an email including an activity do not necessary corresponds to its performer.
- The notion of ***speech act*** is additionally defined to capture the purpose behind including an activity in an email. In this work, four speech acts are considered:

- *Request act:* The employee requests the execution of an activity from the recipient(s), for example, for the activity *'schedule interview'*, the employee writes: *'can you schedule an interview with this candidate?'* ;
- *Request information act:* The employee requests some information concerning the execution of an activity (status, useful data), for example, for the activity *'forward resume'*, the employee writes *'who is the best person in london to forward the resume to?'*;
- *Intention act:* The employee expresses an intention of doing the activity in the future (by himself or other actors), for example, for the activity *'schedule a meeting'*, the employee writes; *'kate, my assistant, will schedule a meeting'*;
- *Information act:* The employee uses the email for informing about activity execution status (it was executed or not or in current execution), for example, for the activity *'create deal'* the employee writes; *'I created the deal'*;

In what follows, let $\mathcal{SA}$ denote the set of speech acts and let $\mathcal{ASA}$ denote the set of activities specified by speech acts. We mean by an activity AN specified by a speech act SA, the concatenation between AN and SA having the following format: AN_SA. Based on these notions, we define an event type as follows (Definition 4.6):

Definition 4.6. ***(Event Type)*** *Let $ev \in EL$ be an event as defined in Definition 4.5 such that it corresponds to the occurrence of an activity $AN \in \mathcal{AN}$ with the speech act $SA \in \mathcal{SA}$. Let $\mathcal{P}$ denote the set of event perspectives such that $\mathcal{P}$=$\{'ActivityName', 'SpeechAct'\}$. Let $f_{projection} : EL \times \mathcal{P}^* \rightarrow \mathcal{SA} \cap \mathcal{ASA}$ be the function that projects an event on a given set of perspectives. An event type related to the event ev w.r.t a set of perspectives $Pers \in \mathcal{P}^*$ is formally defined as $ev_{type} = f_{projection}(ev, Pers)$ where:*

$$f_{projection}(ev, Pers) = \begin{cases} AN & \text{if } Pers = \{'ActivityName'\} \\ SA & \text{if } Pers = \{'SpeechAct'\} \\ AN\_SA & \text{if } Pers = \{'ActivityName', 'SpeechAct'\} \end{cases}$$

An event type refers to from which perspective an event can be viewed. It could be: (i) a speech act, (ii) an activity, or (iii) an activity specified by its speech act (i.e. a combination

between the activity and its speech act) where the event is viewed from activity and speech act perspectives simultaneously. Three event types could be then inferred from the occurrence of one activity (illustrated in our metamodel through the association *'+induces'* between the entities *'ActivityOccurrence'* and *'EventType'*). Taking the example of the activity *'purchase a product'* occurring with request speech act in an email, the inferred event types will be as follows:

- *'purchase product'* of activity event type (i.e. the event is viewed from activity perspective);
- *'request'* of speech act event type (i.e. the event is viewed from speech act perspective);
- *'purchase product_request'* of event type: activity specified by its speech act (i.e. the event is viewed from activity and speech act perspectives simultaneously). It refers to the event of requesting the purchase of a product.

4.5 Output: BP fragments w.r.t functional, organizational, data & behavioral perspectives

A BP fragment in our work is discovered with respect to four perspectives: (1) Functional Perspective (see Section 4.5.1), (2) Organizational perspective (see Section 4.5.2), (3) Data perspective (see Section 4.5.3), and (4) Behavioral perspective (see Section 4.5.4). We further detail in what follows each one of these perspectives.

4.5.1 Functional perspective

The functional perspective of a BP fragment defines the set of activities assigned to it, which refer to the atomic business goals that it aims to realize. This set of activities reflects a partial execution of one BP by a group of actors. A group of actors at this level refers to a set of interlocutors that frequently exchange emails.

We adopt the notion of 'BP fragments' rather than complete 'BP' in view of the incompleteness of BP traces in emails. In fact, there are no restrictions applied on the amount of BP knowledge that must/could be included in emails when performing each BP instance. Hence, for the same BP, performed activities through emails could vary from one instance to another. For instance, in the case of BP implemented in existing management systems, employees could additionally use emails to: (i) Introduce problems regarding BP activities execution and perform additional activities judged necessary to handle these problems, (ii) Perform additional activities considered important to perform BP parts not implemented in a BP management system, and (iii) Tell / Explain how a complete/partial business objective can be achieved.

4.5.2 Organizational Perspective

The organizational perspective of a BP fragment refers to the set of actors intervening, with different contributions, for performing each activity. This organizational perspective is centered on four main notions : (i) Activity actor, (ii) Activity actor group, (iii) Activity Sender-Receivers group and (iv) Actor contribution.

We formally define an activity actor as follows (Definition 4.7):

Definition 4.7. ***(Activity Actor)*** *An actor At of an activity Act* $\in \mathcal{A}$ *is an email user that has received or sent emails containing an activity Act. It is defined as* $At = (AD, OR, BR)$ *such that:*
- $AD \in \mathcal{AD}$ *is the actor email address;*
- $OR \in \mathcal{OR}$ *is the actor organizational role;*
- $BR \in \mathcal{BR}$ *is the actor business role;*

Let $\mathcal{AT}$ denote the set of activity actors. An **actor group** of an activity $Act \in \mathcal{A}$ is defined as a set of actors ($\subset AT$) frequently interchanging emails related to the activity *Act*. In what follows, $\mathcal{G}_{At}$ will denote the set of actor groups of activities. As for the activity Sender-Receivers groups, we formally define them as follows (Definition 4.8):

Definition 4.8. ***(Activity Sender-Receivers group)*** *Let* $Act \in \mathcal{A}$ *an activity and let* $G_{Act} \in \mathcal{G}_{At}$ *be the set of actor groups of Act. A Sender-Receivers group of the activity Act is defined as* $G_{sr} = (sender_{sr}, G_{receivers}, L_{SA})$ *such that;*

- $sender_{sr} \in G_{Act}$ *is an actor frequently sending emails related to the activity Act;*
- $G_{receivers} \subset G_{Act}$ *is the set of actors that frequently coexist in the receivers list of the emails sent by* $sender_{sr}$ *and related to the activity Act;*
- $L_{SA} \subset \mathcal{SA}$ *is the set of speech acts used by* $sender_{sr}$ *when sending activity related emails to* $G_{receivers}$*;*

A Sender-Receivers group of an activity refers to a set of actors frequently receiving activity related emails from the same sender with a specific list of speech acts. This set of actors must coexist in the same receivers' list of these emails. If two sets of actors receive separate emails in the context of the same activity from the same sender, they will form two different Sender-Receivers groups. Taking the example of one project manager S_{PM}, he/she could send emails to his/her team members G_{team} to request or plan the execution of one activity Act_p. He/she could equally send other emails to his/her supervisors $G_{supervisors}$ to inform about the activity execution status. In such case, as illustrated in Figure 4.3, $G_{sr_1} = (S_{PM}, G_{team}, \{$*'request', 'intention'*$\})$ forms one Sender-Receivers group and $G_{sr_2} = (S_{PM}, G_{supervisors}, \{$*'information'*$\})$ forms another Sender-Receivers group.

Such notion of Sender-Receivers group presents an elementary entity to globally reflect how emails are exchanged by actors in the context of the same activity. As one email could be

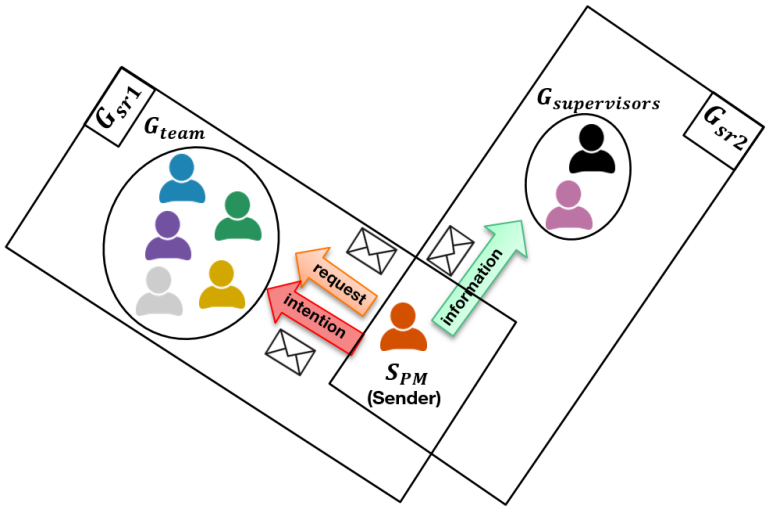


Figure 4.3: Example of Sender-Receivers groups interactions

sent by the same sender to more than one receiver, we adopt this notion to synthesize email exchanging between actors and to avoid information redundancy (e.g. in the context of the activity Act_p, G_{sr_1} synthesizes email exchanging between S_{PM} and his team members G_{team} rather than presenting it through a set of relations between S_{PM} and each team member).

An **actor contribution** defines the type of intervention made by him/her towards performing an activity. It is of six types which were identified based on our analysis of industrial emails and discussions with BP experts:

- *Request contribution:* It refers to requesting the execution of an activity from other actors;
- *Request Information:* It refers to requesting some useful information concerning the execution of an activity (e.g., status, useful data, opinion, permission);
- *Execution contribution:* It refers to performing the activity;
- *Information contribution:* It refers to informing about the execution of an activity (executed or not);
- *Planning contribution:* It refers to planning how and by whom activity will be performed.
- *Observation contribution:* It refers to observing how an activity is performed, without doing a concrete action towards its execution;

In what follows, let ζ denote the set of contributions and let $\mathcal{G}_{SR}$ denote the set of Sender-Receivers groups of activities. The overall organizational perspective of an activity is formally defined as follows (Definition 4.9):

Definition 4.9. ***(Activity Organizational Perspective)*** *The organizational perspective of an activity $Act \in \mathcal{A}$ is defined as a tuple $\Omega_{Act} = (S_{AT}, \mathcal{F}, \mathcal{F}_{SR}, R, C)$ where;*

- $S_{AT} = \{At_1, .., At_n\} \subset \mathcal{AT}$ is the set of n actors;
- $\mathcal{F} : S_{AT} \rightarrow \mathcal{G}^*_{At}$ is the function that maps each actor to his/her groups (one actor can be affected to multiple groups);
- $\mathcal{F}_{SR} : S_{AT} \rightarrow \mathcal{G}^*_{SR}$ is the function that maps each actor to his/her Sender-Receivers groups (one actor can be affected to multiple Sender-Receivers groups);
- $C = \{\sigma_1, \ldots, \sigma_n\}$ is the set of distributions of actors' contributions towards the activity act. Each $\sigma_i = \{(c, \lambda_{c,i}) \mid c \in \zeta \wedge \lambda_{c,i} \in [0, 1]\}$ refers to the distribution of the contributions ($c \in \zeta$) of actor At_i towards the occurrences of the activity Act such that $\sum_c \lambda_{c,i} = 1$. A tuple $(c, \lambda_{c,i}) \in C_i$ refers to a contribution c and its fraction $\lambda_{c,i}$;

For an activity, the organizational perspective defines the set of actors and actors groups intervening for performing it. For each actor, we discover the distribution of his/her overall contributions towards performing all activity occurrences. This distribution is defined according to the coefficients $\lambda_{c,i}$. The closer $\lambda_{c,i}$ is to 1, the more its contribution c is dominant. Figure 3.1 (c) represents an example of visualization (i.e. pie-chart) of such distribution. The organizational perspective equally identifies the global interactions made by senders and receivers in the context of the same activity. These interactions are discovered in the form of *Sender-Receivers groups* which characterize the operation of sending emails from a sender to a set of receivers by the related speech acts.

4.5.3 Data perspective

The data perspective of a BP fragment is defined through:

- The data schema of the business information introduced in e-mails. It is discovered in the form of a set of artifacts having association relations between each others and specified by a set of cardinalities (these relations are illustrated in our meta-model through the self association *+has association with* of the entity *'Artifact'*). These artifacts organize business information into informational entities characterized by a set of data attributes. For more details, see Section 7.2 for artifact discovery and 7.4 for mining artifact cardinalities;
- The artifacts manipulated by BP fragment activities. This is discovered in the form of a set of associations between activities and artifacts (This is illustrated in our meta-model through the association *+manipulates* between the entities *'Activity'* and *'Artifact'*;).

Our data perspective is then centered on three notions: (1) **Artifact**, (2) **Association** and (3) **Artifact cardinality**.

An artifact is an entity mapped to a set of business information $BI \subset \mathcal{BD} \cup \mathcal{BC}$ and that receives the action of a set of activities (i.e. at least one activity). It is composed of: (1) a name, and (2) set of attributes. Taking the example of a trading deal artifact, three business information could be mapped to it which are; 'deal identifier', 'deal price' and 'deal counterparty'.

An association is a coexistence relation, deduced from email communication traces, between: (1) Artifact-Artifact to refer to what types of artifacts have common business context, or (2) Artifact-Activity to refer to what artifacts are manipulated by which activities.

Each Artifact-Artifact association is, as illustrated in Figure 4.4, specified by an **artifact cardinality** $card = (m_{12}, m_{21})$ composed of two multiplicities m_{12} and m_{21}. Each multiplicity m_{ij} $(i \neq j)$ denotes the number of instances of one artifact $i \in \{1, 2\}$ that can occur on a given instantiation of the other artifact $j \in \{1, 2\}$. We adopt three multiplicity types (m_{ij}) as follows:

- **One** multiplicity denoted by **1**; in the case of exactly one instance of Artifact i is induced by the instantiation of Artifact j, for example, a contract is associated to exactly one trading deal;
- **Many** multiplicity denoted by **1..***; in the case of at least one instance of Artifact i is induced by the instantiation of Artifact j, for example, a customer could perform multiple trading deals;
- **Zero-or-One** multiplicity denoted by **0..1**; in the case of no instances or one instance of Artifact i is induced by the instantiation of Artifact j, for example, one deal could be associated to zero or one transport contract (zero in the case of a cancelled deal);

Figure 4.4: Artifact Association

Taking the example of Figure 3.2, an artifact-artifact association is illustrated between the artifacts 'Gas Meter' (Artifact1) and 'Gas Deal' (Artifact2) with multiplicities $m_{21} = 1$ and $m_{12} = 1..*$

In what follows, let $\mathcal{AR}$ be the set of artifacts, $\mathcal{AS}$ be the set of associations and $\mathcal{ARC}$ be the set of artifact cardinalities. A data perspective of a BP fragment is formally defined in our work as follows (Definition 4.10):

Definition 4.10. ***(BP fragment Data Perspective)*** *Let $A_{PF} \subset \mathcal{A}$ be the set of activities of a BP fragment PF. Data perspective of PF is defined as $\mathcal{D}_{Per} = (A_{PF}, S_{Ar}, Ass, cards) \in \mathcal{A} \times \mathcal{AR} \times \mathcal{AS} \times \mathcal{ARC}$, where;*
- $S_{Ar} = \{Ar_1, Ar_1, .., Ar_m\} \subseteq AR$ is a set of m artifacts.
- $Ass \subseteq \mathcal{AS}$ is the set of associations between pairs of artifacts ($\subseteq S_{Ar}$), activities ($\subseteq A_{PF}$) or artifacts and activities ($\subset S_{Ar} \times A_{PF}$).
- $cards \subseteq \mathcal{ARC}$ is the set of artifacts cardinalities of artifact-artifact associations ($\subseteq Ass$).

4.5.4 BP fragment Behavioral Perspective

The behavioral perspective defines the control flow conditions in a BP fragment. We define these conditions w.r.t each event type (i.e. speech act, activity or activity specified by its speech acts, see Definition 4.6) through a set of sequencing constraints. These sequencing constraints are discovered to obtain three model types for each BP fragment: (i) **Activity model**, (ii) **SA model**, and (iii) **Activity & SA model**. In what follows, we formally define a sequencing constraint in Section 4.5.4.1. Then, we formalize the definitions of our three model types (Section 4.5.4.2, Section 4.5.4.3 and Section 4.5.4.4). We remind that $\mathcal{A}$, $\mathcal{SA}$ and $\mathcal{ASA}$ denote respectively the set of activities, speech acts and activities specified by their speech acts.

4.5.4.1 Sequencing constraints

We define a sequencing constraint through the following definition (Definition 4.11):

Definition 4.11. ***(Sequencing Constraint)*** *A sequencing constraint* $Const = (ref, targ, const)$ *is defined by a reference event type* $ref \in \mathcal{A} \cup \mathcal{SA} \cup \mathcal{ASA}$*, a target event type* $targ \in \mathcal{A} \cup \mathcal{SA} \cup \mathcal{ASA}$ *and a constraint type* $consttype \in \mathcal{CT}$ *where* $\mathcal{CT}$*={'response', 'co-existence'} defining the relation between the reference and the target event types. It is denoted as:* $ref \xrightarrow{consttype} targ$

A **Sequencing constraint** is of two types of behavioral constraints between a reference event and a target event:

- *Response constraint*: It defines the case where the occurrence of an event type (reference event type) induces as a response the occurrence of another event type (target event type). In such case, the reference event type uses to occur before the target event type (temporally).
- *Coexistence dependency constraint*: It defines the case where the occurrence of an event type (reference event type) induces the appearance of another event type (target event type) in the same email.

4.5.4.2 Activity model

We define the BP activity model of a BP fragment as follows (Definition 4.12):

Definition 4.12. ***(Activity Model)*** *An Activity model of a BP fragment is defined as* $\mathcal{M}_{Act} = (A_{PF}, C_A)$*, while;*
- $A_{PF} \subset \mathcal{A}$ *is the set of BP fragment activities;*
- $C_A \in A_{PF} \times A_{PF} \times \mathcal{CT}$ *is the set of activity constraints;*

An activity model illustrates the sequencing constraints between activity event types. Taking the example of the activity model in Figure 4.5, it illustrates the following sequencing constraints: *'flow deal'* $\xrightarrow{response}$ *'roll deal'*; *'flow deal'* $\xrightarrow{response}$ *'extend deal'*; and *'flow deal'* $\xrightarrow{response}$ *'create deal'*. An activity model actually reflects :
- How BP fragments are described by employees themselves in the same emails (e.g. email in Figure 1.3): this is the case where employees use emails as support to explain to other employees what actions have induced the necessity of performing other actions in the future, or what activities must be performed to achieve a business goal.
- What activities are used to be performed as a response to event occurrences (mentioned in separate emails, e.g. emails in Figure 1.1). This is the case where a set of employees interchange emails to inform about the occurrence of one event and what activities were or to be done as a response.

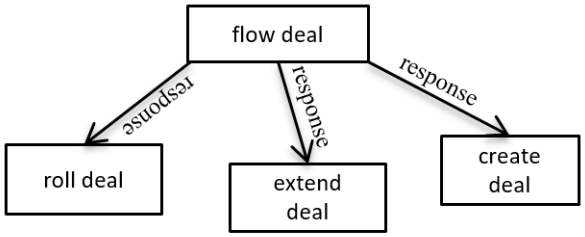


Figure 4.5: Activity model example

4.5.4.3 SA model

We define the SA model as follows (Definition 4.13):

Definition 4.13. ***(SA Model))*** *A SA model of an activity Act* $\in \mathcal{A}$ *is defined as* $\mathcal{M}_{Act-SA} = (Act, S_a, C_{SA})$*, while;*
- $S_a \subset \mathcal{SA}$ *is the set of activity speech acts;*
- $C_{SA} \in S_a \times S_a \times \mathcal{CT}$*: is the set of speech act constraints;*

A SA model of an activity *Act* illustrates the sequencing constraints between speech act event types of the same activity *Act* when interchanging emails between employees in order to perform it. Taking the example of the SA model in Figure 4.6, it illustrates the following sequencing constraints: *request* $\xrightarrow{response}$ *information*. It shows how employees could interchange emails containing a 'request' and then an 'information' of the same activity 'extend deal'. This kind of models would be useful for estimating the average time needed for performing an activity (e.g. the average time needed to pass from 'request' status to 'information' status).

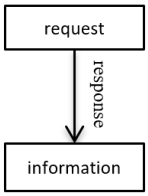


Figure 4.6: SA model of 'extend deal' activity

4.5.4.4 Activity & SA model

We define the Activity & SA model as follows (Definition 4.14):

Definition 4.14. ***(Activity** & **SA BP Model))*** *Activity & SA model of a BP fragment is defined as $\mathcal{M}_{ASA} = (ActSA, C_{ASA})$, where;*
- $ActSA \subset \mathcal{ASA}$ is the set of specified activities;
- $C_{ASA} \subset ActSA \times ActSA \times \mathcal{CT}$ is the set of specified activities constraints;

An Activity & SA BP model illustrates the sequencing constraints between activities specified by their speech acts when exchanging emails between employees. It reflects how a BP fragment is observed through emails. This includes how emails are used w.r.t activities and how the speech act sequencing of different activities could be intercepted for realising a BP fragment. Taking the example of the activity & SA model in Figure 4.7, it visualizes the

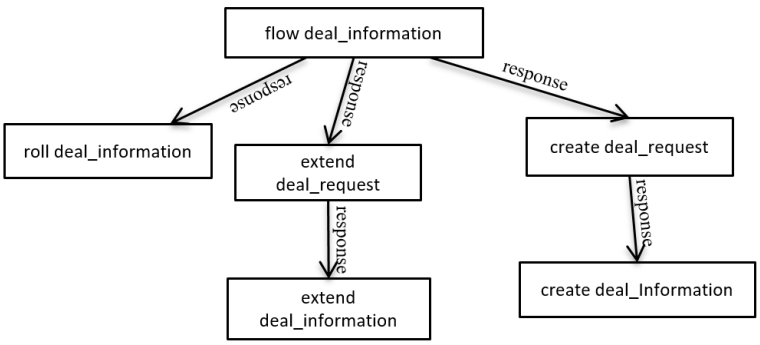


Figure 4.7: Activity & SA Model

following constraints:
- 'flow deal_information' $\xrightarrow{response}$ 'roll deal_information';
- 'flow deal_information' $\xrightarrow{response}$ 'extend deal_request';
- 'flow deal_information' $\xrightarrow{response}$ 'create deal_request';
- 'extend deal_request' $\xrightarrow{response}$ 'extend deal_information';
- 'create deal_request' $\xrightarrow{response}$ 'create deal_information';

4.6 Case Study

In this section, we demonstrate the validity of some design choices adopted in our approach meta-model. We use a real-life email dataset composed of 1103 emails collected from Orange and Enron employee mailboxes. We start by a description of the dataset and how it was collected (Section 4.6.1). Then, we justify our model design choices through: (i) some concrete examples (Section 4.6.2) and, (ii) one experiment where we use the overall collected dataset (Section 4.6.3).

4.6.1 Dataset Collection Method and Description

To perform our experiments, we have collected a set of emails related to different BP performed within Enron and Orange Labs.

For Enron emails, we were based on the publicly available dataset Enron[1]. We have selected 674 emails from the outboxes of two employees having different organizational and business roles and involved in different BP. Therefore, we have annotated each email in terms of the occurred activities and their related speech acts. Table 4.2 summarizes the features of the annotated emails. For each employee, it denotes the number of its annotated emails, his/her organizational and business role and the names of the BP related to the annotated activities.

Table 4.2: Statistics on the selected Enron data

ID	# emails	Organizational Role	Business Role	BP names	Start	End
1	342	Assistant	Risk Management	Hiring Setting, Meeting Setting, Conference Participation	Dec 10, 1999	Jan 9, 2002
2	332	Manager	Trading	Trading of Electricity Power	May 30 , 2001	Jan 31, 2002

As for Orange emails, we have used an email client (Microsoft Outlook) plugin installed in three employees working environment to collect them. The employees have annotated their emails as they received them by specifying (i) the related BP name (e.g. Hiring process, patent application process, etc.), and (ii) the related BP activity names (e.g. interview wrapup, decision notification, patent description writing, etc.). Additionally, the same plugin have collected for each email the following fields: the sender of the email, the recipients, the copied (cc and bcc fields), the subject, the content (including the previous messages of the same conversation) and the time at which the email was sent or received. We refer to the dataset collected in this phase as *raw dataset*. The raw dataset was enriched with additional

[1]http://www.cs.cmu.edu/~enron

attributes based on the proposed metamodel. These attributes are: activity occurrences speech acts and the related actors contributions. The collected data contains several BP, but in this paper we will focus only on three BP; Hiring, paper submission and patent application BP resulting in 429 emails. Table 4.3 summarizes the features of related data.

Table 4.3: Statistics on the selected Orange data

BP name	**# emails**	**# BP instances**	**# activities**	**Start**	**End**
Hiring	220	3	49	Jan 3, 2018	Jan 1, 2020
Paper Submission	167	6	25	Jan 9, 2018	Jan 31, 2020
Patent Application	42	1	11	Jan 11, 2019	Jan 31, 2020

4.6.2 First validation

In this section, we use two concrete examples to validate the design choices made in the proposed metamodel, specifically in the *Event Log* and the *Final Output* parts.

Figure 4.8 shows an example of an email containing six different coherent expressions. Each one refers to one activity occurrence. The first coherent expression is introductory and refers to a specific instance of the hiring process; the instance related to "thesis in decisional/transactional systems" position, which represents the job position title. The job position is actually an artifact in the hiring process and the title is one of its data attributes. The second coherent expression informs about the status of select candidate activity. The third expression is asking for the status of a specific activity (i.e. *study candidature*) in a BP fragment within the hiring BP. The fourth expression recalls an activity (i.e. *send candidature*) that was done in this context. The fifth expression informs the recipients about the status of a job position description "the post-doctoral subject", which represents a title attribute instance of the job position artifact (of the hiring process). Finally, the sixth expression reveals an intention about sending the job position description.

The above example confirms and consolidates several choices and hypothesis made in the proposed approach metamodel. First, some emails may contain several *coherent expressions*. They would refer to different activities and different speech acts. The illustrated example contains a "request" for asking for status and "information" about a post-doctoral offer description. Second, a *coherent expression* may refer to the occurrence of a *business information occurrence*. In the illustrated example, we can distinguish several business data occurrences: (1) "thesis" which is a value of the business information "position type", (2) "decisional/transactional systems" which is a value of the business data "position title", (3) "first name" and "last name" which are values of "first name" and "last name" business data of the artifact candidate, and (4) "the post-doctoral subject" which is a value of the business data "position title" (of the artifact "Position").

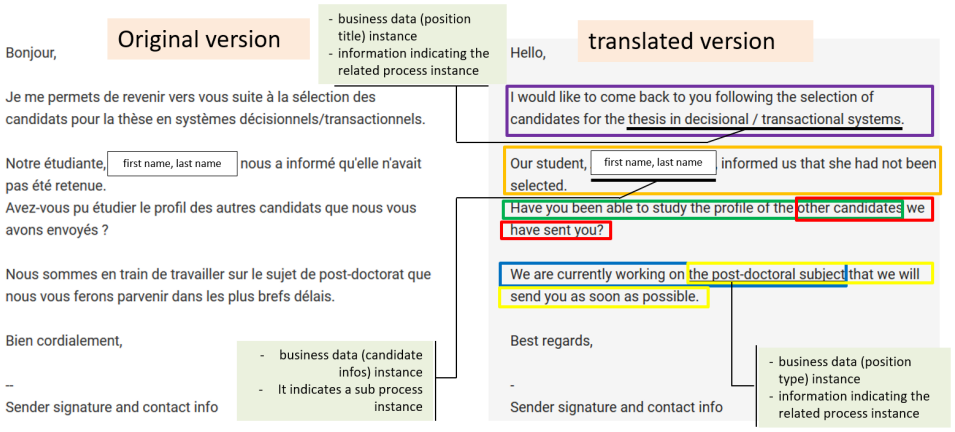


Figure 4.8: Email example from Orange dataset (translation source: http://translate.google.com)

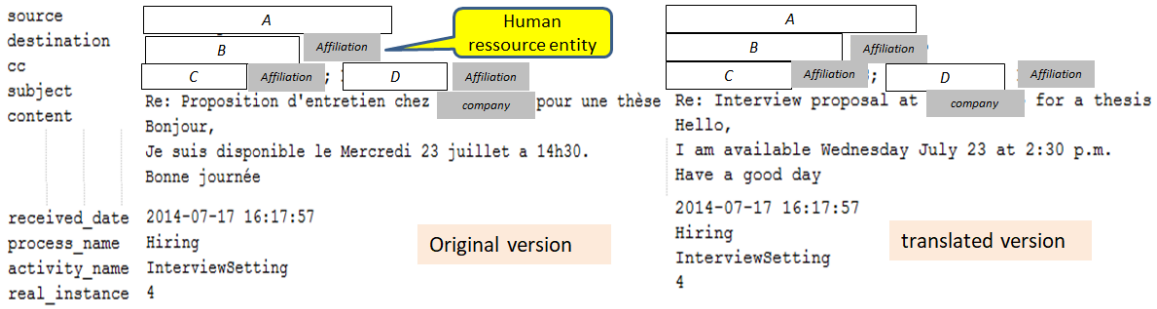


Figure 4.9: Email example showing the organizational perspective (retreived from Orange dataset)

In order to illustrate the *Actor* and *Contribution* aspects in the proposed event log metamodel, let's consider the email displayed in Figure 4.9. It executes an "interview setting" activity of the hiring process. It is sent by candidate (A) to an HR (Human Resource) interviewer (B). The copied persons of this email (C and D) are part of the recruiting team (R&D). They do not take part of the interview but, still, they are informed about the execution of the "interview setting" activity. This shows that the actors referred in an email related to a process may have different contributions in its execution and in the execution of a specific activity. In this example, actors A and B are "executors" of the activity and C and D are "observers" of the execution. This validates different assumptions: (1) several actors are involved in an activity occurrence, and (2) the actors may have different contributions in an activity occurrence.

4.6.3 Second validation

In this experiment, we use our overall dataset to show that the main design choices made in our metamodel work in a reasonable number of real emails. These design choices concern:

- **DesignChoice 1:** The multiplicity of activity component occurrences in one email;
- **DesignChoice 2:** The use of emails for multiple purposes;

- **DesignChoice 3:** The multiple actor contributions made in the context of one activity

For the first design choice, we proceeded as follows to validate it. First, we calculated for each email the number of annotated activities. Then, we calculated the distribution of activity numbers over emails. Figure 4.10 and Figure 4.11 shows the obtained results for the Orange and Enron datasets respectively. As noticed, around 74% of emails in the Orange dataset contained one activity, the rest contained more than one activity. As for the Enron dataset, only 35% of emails contained one activity, around 52% of emails contained multiple activities while the number goes from 2 to 14 (the high number of activities appears especially at the level of long emails sent by the trading manager to plan future deals). These results prove the importance of considering the multiplicity of activities in emails in our approach metamodel and in our solution for activity discovery. Finally, we can notice that around 12% of enron emails have not been assigned to any annotated activity. This show that not all emails are BP oriented, which justifies the cardinality *'0..1'* relating the entities *'ActionExpression'* and *'ActivityNameOccurrence'* in our approach metamodel (see Figure 4.1). It is important to note that all emails in the Orange dataset are BP oriented (unlike those of the Enron dataset). In fact, employees generally select emails containing BP activities in the annotation process. This is not the case of Enron emails, as for constructing our dataset, we selected all emails belonging to the outbox of two employees without filtering them.

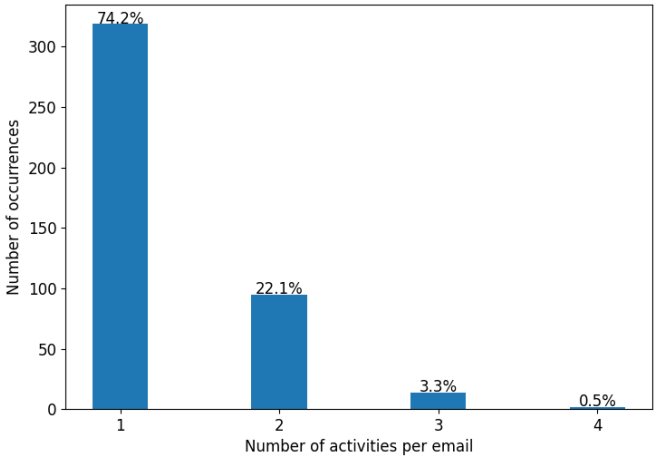


Figure 4.10: Distribution of the number of activities per email in the Orange dataset

For validating the second design choice, we calculated the distribution of activity occurrences' speech acts over the Enron and Orange datasets. We recall that these speech acts reflect the senders' purpose when including activities in their emails. Figure 4.12 outlines the obtained results. We can notice that around 43% of speech acts of the occurred activities have appeared in emails with *'information'* speech act. The rest have appeared with *'intention'* (27.2%), *'request'* (28.3%) or *'request information'* (1.1%) speech acts. This shows the importance of considering the fact that the use of emails in the context of BP activities could cover multiple purposes.

For validating the third design choice, we have additionally annotated the Orange dataset

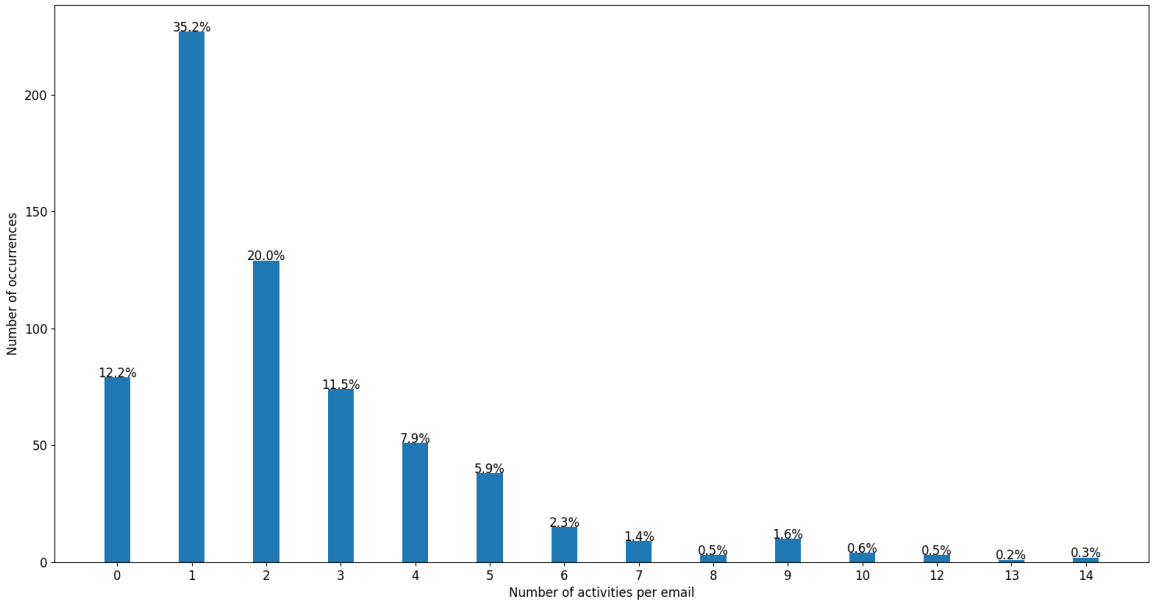


Figure 4.11: Distribution of the number of activities per email in the Enron dataset

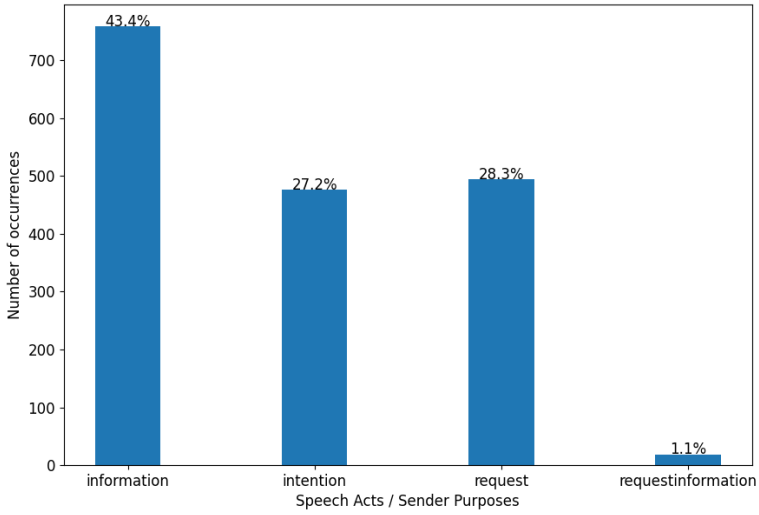


Figure 4.12: Speech acts distribution in Orange and Enron datasets

in terms of actor contributions. During the annotation process, we have manually deduced these contributions from: (i) the activity occurrences in emails, and (ii) their related speech acts. The bar chart in Figure 4.13 shows the distribution of these contributions where the bars reflect the numbers of occurrences of each manually assigned contribution. This bar chart reveals that more than 63% of the assigned contributions are different from the execution contribution. This proves that actors' contributions are not limited to executing activities when interacting using emails.

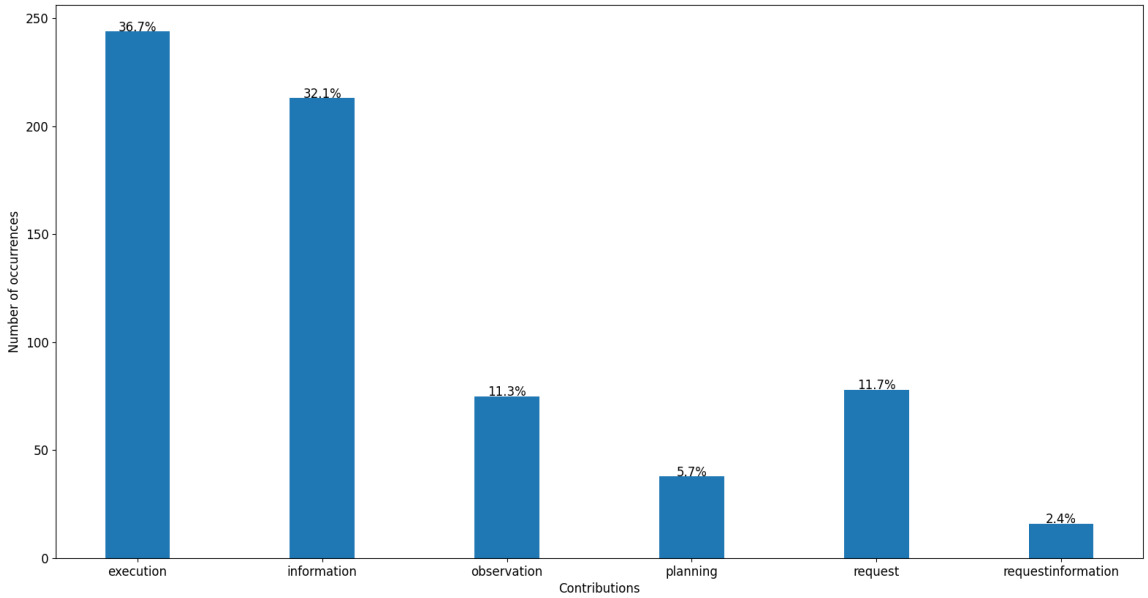


Figure 4.13: Distribution of actors' contributions in Orange dataset

4.7 Conclusion

In this chapter, we answered the first question (**Q1**) raised in the thesis problematic (Section 1.2.1), which is : What are the BP knowledge (i.e. BP perspectives and BP elements) that could be discovered from emails and how to formalize them ? and its three sub-questions which are:

- Q1-1: What are the BP perspectives that could be discovered from emails while considering the specificities of using emailing systems ?
- Q1-2: What is the event log structured enabling the discovery of the defined BP perspectives ? i.e. What are the additional/less BP elements (compared to those of the most common event logs) to be retrieved from emails ?
- Q1-3: How to define the actual BP events related to activity occurrences (i.e. appearance) in emails taking into account the specifics of emailing systems ?

To answer the first and the second sub-questions (**Q1-1**, **Q1-2**), we carried out an exploratory task and we identified four BP fragments perspectives to be discovered from emails: functional, data, actors and behavioral perspectives. We formalized the definition of these perspectives and their relation with the unstructured log data of emails. We additionally described such relation through a metamodel that outlines the main entities ensuring the transformation of an email log into valuable BP knowledge. At the level of the intermediate part of such metamodel, we formally defined an event log structure consistent with emailing systems specifities. We also outlined the main entities that ensure the transformation of email log data into such structured format.

To answer the third sub-question (**Q1-3**), we defined three event types that take into account the activity and the speech act dimensions.

In the next chapters, we will focus on detailing our algorithmic approaches that we have introduced to automate the discovery of the different formalized BP knowledge in this chapter (see Chapter 6, Chapter 7).

Chapter 5

First Step Towards Event Log Generation: Unsupervised Learning For Activity Discovery

Contents

5.1 Introduction

Event log presents a crucial input enabling BP discovery. As the raw email log has not the suitable structure, it must be converted into a structured event log to ensure BP discovery. As mentioned earlier, an important step to the generation of such event log is to recognize BP activities to enable the discovery of their occurrences in emails and to capture the related event attributes.

To achieve this goal, as stated in Chapter 2, different approaches were proposed [55, 52, 77, 18, 43, 76, 86, 48]. However, most of them relied on integrating supervised learning techniques [18, 55, 76, 86]. This requires human intervention (e.g. for labeling training dataset) and needs knowing activities in advance, which is not always feasible. As for approaches which integrated unsupervised techniques as [48], they considered a sentence as the lowest structure that could express an activity. Actually, in the case of non-controlled systems as emails, the expression of activities would not be constrained by emails' punctuation; employees could express more than one activity in the same sentence or email. Finally, the same approaches focused on discovering activities without considering the associated business information.

Given these limitations, we introduce in this chapter a completely unsupervised learning solution for discovering activities from emails as a first part of the event log generation phase. In the design of our solution, we make two main assumptions:

- **Assumption 1**: Activities characterizing BP tend to be frequent in BP traces. For instance, performing different executions of the same BP leads to repetitively realizing the same set of activities. This leads to a frequent appearance of these activities in the corresponding execution traces. Therefore discovering frequent activities as a first step could bring us to the discovery of BP fragments;
- **Assumption 2**: One employee that frequently expresses the same activity in his/her emails would probably use close expressions. Therefore, discovering these frequent expressions from emails would be a first step towards the discovery of BP activities;

Based on these assumptions, we propose to discover frequent activities from emails. We introduce for this purpose a solution based on pattern discovery to mine frequent expressions used by employees to express their activities. This approach discovers each activity as the composition of activity name, business data and business context information. It learns a model that associates to each activity component a set of patterns frequently used by BP actors to express it. Our approach consists of two main steps: The first one analyzes per employee emails to reduce expression variance. It aims to capture the set of frequent patterns used by employees in order to express frequent activities. The second step regroups similar activities of different employees. To regroup patterns belonging to the same activity (first step) and to regroup similar activities of different employees (second step), two similarity measures are used on the basis of: (i) words' synonyms, and (ii) activity/pattern business context (defined by their related business information).

Pattern discovery from textual data is a text mining technique that uncovers relevant frequent substructures (e.g., set of words) that co-occur frequently. If one activity is discovered in the form of the set of patterns that are frequently used to express it, it would be detected more flexibly without being constrained by emails' punctuation. Based on recent surveys in such context [62], Sequential Patterns Mining (SPM) techniques present the state of the art today (e.g; n-gram generators, PrefixSpan). They identify words while paying attention to the relationships between them so that their semantic meanings can be preserved. However, they are based on the sequencing of words during the construction of patterns. For this

reason, they are likely to miss the detection of important activities since in email bodies; words expressing the same activity (i) may not be identical, (ii) may not follow same order, and (iii) are not necessarily adjacent in sentences.
In view of these shortcomings, we introduce an approach for pattern discovery that allows the discovery of common substructures of activities while tolerating using words that could be: (i) different but sharing the same meaning, (ii) discontinuous, and (iii) not sequential.

To evaluate the overall unsupervised learning solution for activity discovery, we rely on a real dataset of emails retrieved from the public dataset Enron[1]. This dataset reports emails sent or received by Enron employees operating in the field of online trading of energy. We demonstrate the effectiveness of our solution by studying the recall and the significance of the discovered activities. We additionally show examples of results obtained after analysing real emails of Orange employees.

In the following (see the framed phase/part with the orange color in the framework extract illustrated in Figure 5.1), we start in Section 5.2 by explaining the first phase in our overall framework that pre-processes email log data. Then, in Section 5.3, we present each step in the unsupervised learning part for activity discovery from emails (i.e. Part 2.1 in Figure 5.1). Finally and before concluding, we evaluate our proposals in Section 5.4.

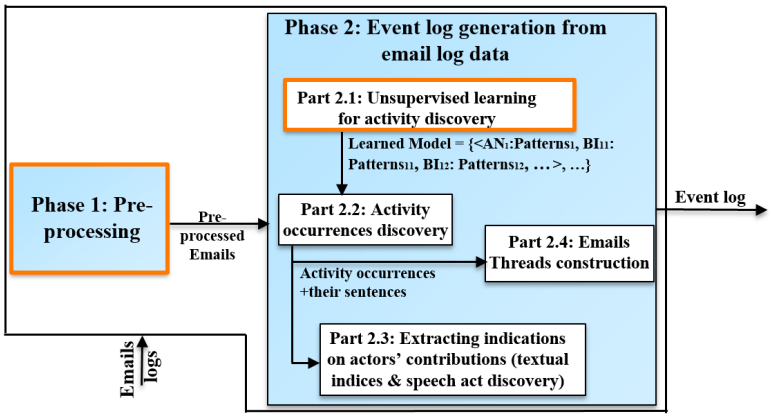


Figure 5.1: Framework extract: Phase 1 & Part 2.1

The main contributions of this chapter were published in the forum of the Business Process Management (BPM) conference [29].

5.2 Email log preprocessing

This section focuses on the first phase in our framework, which preprocesses email log data (EmL, Definition 4.1). It applies natural language processing operations on the main body and the subject of each email $email \in EmL$ to return a list of tuples mapping each word to some local features (e.g. part of speech tag, position of appearance in email). This phase

[1]https://www.cs.cmu.edu/~enron/

applies four main steps as depicted in Figure 5.2:

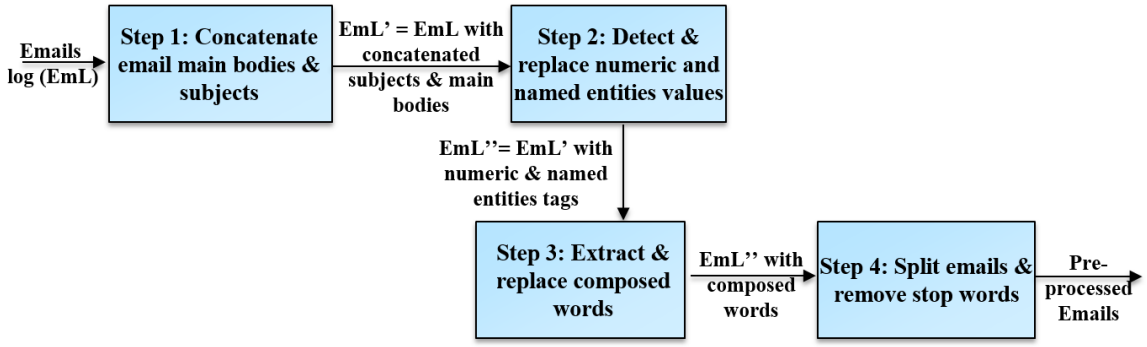


Figure 5.2: Main Steps of the preprocessing phase

Step 1: Concatenate email main body and subject: For each email, this step concatenates the main body and the subject into one email string; it adds the subject as a sentence at the beginning of the main body. To avoid considering the subject of the same conversation several times, it is only added at the beginning of the main body if the corresponding email is used to start the conversation. In our case, this is identified by checking the textual contents of the email subjects to see if they contain reply or forward indicators (i.e. *'Re:'*, *'Fwd:'*).
Step 2: Detect numeric and named entities values: This step detects numeric values and named entities and it replaces them, in the email string, by tags reflecting their types. Six types are considered: Price, date, numeric, person name, organization name and location. Table 5.1 outlines these types and their related tags and descriptions. Such operation is useful for discovering business data patterns and values as numeric and named entities terms that are most often used by employees in this context. For instance, employees would use numeric terms for typing an ID of a deal/purchase order. In a recruitment context, they use person names for mentioning candidate names.

Table 5.1: The types of the detected numeric values and named entities

Type	Tag	Description
Price	pricenumeric	It replaces numeric values expressing prices (e.g. 25$)
Date	datenumeric	It replaces numeric values expressing dates (e.g. 30 January 2000)
Numeric	numeric	It replaces the remained types of numeric values (which are different from the date and the price types)
Person Name	personname	It replaces textual terms used for expressing person names.
Organization Name	orgname	It replaces textual terms used for expressing organization names.
Location	locname	It replaces textual terms used for expressing location names.

Step 3: Extract composed words: This step extracts composed words of non-verbs and replace them in the email string by one term expressing the concatenation of the two words (e.g. the term *'hour0end'* is associated to the composed word 'hour end' after concatenating its two words through the character *'0'*).
Step 4: Split emails & remove stop words: This step first splits each email into sentences using the following punctuation as textual separators: '.', '!', '?'. Then, it splits each sentence into words and removes some stopwords and useless ones (e.g. thanking, salutations,

signatures). Finally, it returns, a structured format, that we define in Definition 5.1, recording words' local features, which will be used to discover activities' patterns.

Definition 5.1. ***(Preprocessed Email)*** *A preprocessed email is a list of tuples while each tuple* $tup = (w_{Rf}, pos, w, t_s, sentPos)$ *is composed of the following elements:*

- *Words' raw form (*w_{Rf}*, e.g. 'created' in 'I created a deal');*
- *Its position of appearance (pos) in the email;*
- *Words' basic or lemmatized form (w), e.g. 'create' is the basic form of 'created';*
- *Words' part of speech tags (*t_s*): only two types of tags are considered: verb ('v') and non-verb ('n') as they are important in defining activity components;*
- *The position of the sentence (sentPos) where words appears in the email;*

For the example of the raw email $email_2$ of Figure 1.1, we remind that it has the following subject and email main body:
- Subject: *'Re:Flow w/ no nom'*
- Main body: *'Rolled deal 454057 to cover flow at mtr 5192'*
After applying the described preprocessing steps, we obtain the following list of tuples after applying this step:
$List_{TUP}$ = [($'Flow'$, 0, $'flow'$, $'n'$,0), ($'nom'$, 1, $'nom'$, $'n'$,0), ($'Rolled'$, 2, $'roll'$, $'v'$, 1), ($'deal'$, 3, $'n'$, 1), ($'454057'$, 4, $'numeric'$, $'n'$, 1), ($'cover'$, 5, $'cover'$, $'v'$, 1), ($'flow'$, 6, $'flow'$, $'n'$, 1), ($'mtr'$, 7, $'mtr'$, $'n'$, 1), ($'5192'$, 8, $'numeric'$, $'n'$, 1)].

In what follows, we denote by $\mathcal{E}$ the set of preprocessed emails. We recall that: (i) $\mathcal{W}$ denote the set of lemmatized words, (ii) $\mathcal{T}$ denote the set of part of speech tag categories. We additionally adopt the following functions:

- $LEM_P : \mathcal{E} \rightarrow \mathcal{W}^*$ is the function that returns for each preprocessed email $em \in \mathbb{E}$, the list of lemmatized words such that $LEM_P[em] = [tup.w,\ tup \in em]$;
- $POSTag_P : \mathcal{E} \times \mathcal{W} \times \mathbb{N} \rightarrow \mathcal{T}$ is the part of speech tag function that returns for a lemmatized word, appearing at a given position $ps \in \mathbb{N}$ in a preprocessed email $em \in \mathcal{E}$, the corresponding part of speech category tag. This means that: $POSTag_P(em, w, ps) = em[ps].t_s$

5.3 Unsupervised Learning part: Activity Discovery

This section focuses on the first part (i.e. Part 2.1 in Figure 5.1) of generating a structured event log from a preprocessed email log. It presents the unsupervised learning solution that we introduce to discover activities from emails. In what follows, we first give an overview on such solution in Section 5.3.1. Then, we detail its main steps in Section 5.3.2 and Section 5.3.3.

In the following, we rely on these two email main bodies' extracts (em_A and em_B) as illustrative examples:
- em_A = *'We shall arrange a preliminary interview for trader position with this person... I have some availability next week; can you contact him to define a time slot?'*
- em_B = *'Another interview should be set with him. Last week a student forum was held and I probably found other potential candidates. I've set up some time slots with them for this week and I will send you my feedback concerning my preliminary interviews with them'.*

For simplicity, we suppose that the beginning of these extracts corresponds to the beginning of the textual content (i.e. concatenation between subjects and main bodies) of the corresponding emails. After applying our preprocessing steps (Section 5.2), we obtain the following results:
- $em_A = [('arrange', 0, 'arrange', 'v', 0), ('preliminary', 1, 'preliminary', 'n', 0), ('interview', 2, 'interview', 'n', 0), ('trader',3, 'trader','n',0), ('position',4, 'position', 'n', 0), ('person', 5, 'n', 0), ('availability', 6, 'availability', 'n', 1),('contact', 7, 'v', 1), ('define', 8, 'v', 1), ('time', 9, 'time', '1), ('slot', 10, 'slot', 'n', 1)]$
- em_B = [('interview', 0, 'interview', 'n', 0), ('set', 1, 'set', 'n', 0), ('student', 2, 'student', 'n', 1), ('forum', 3, 'forum', 'n', 1), ('held', 4, 'hold', 'v', 1), ('probably', 5, 'probably', 'n', 1), ('find', 6, 'find', 'v', 1), ('potential', 7, 'potential', 'n',1), ('candidate', 8, 'candidate', 'v', 1), ('set', 9, 'set', 'v', 2), ('time', 10, 'time', 'n', 2), ('slot', 11, 'slot', 'n', 2), ('send', 12, 'send', 'v', 2), ('feedback', 13, 'feedback',n, 2), ('concern', 14, 'concerning', 'v', 2), ('preliminary', 15,'n','preliminary', 2), ('interview',16, 'interviews','v', 2)].

5.3.1 Overview

As illustrated in Figure 5.3, our learning solution for activity discovery is composed of three main steps :

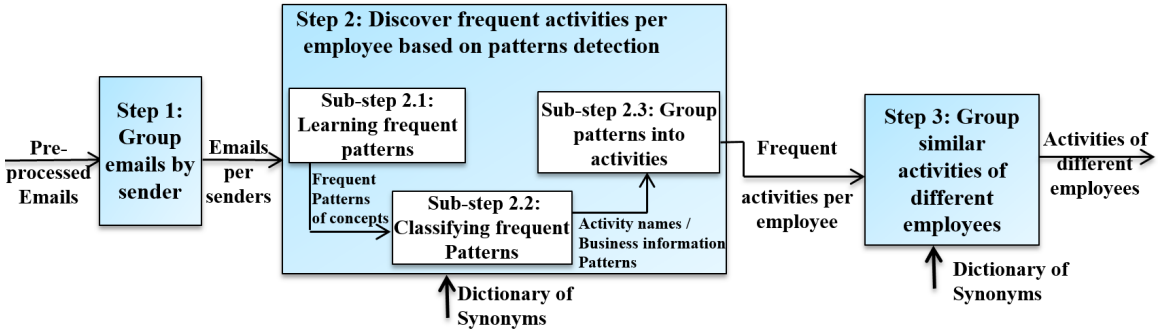


Figure 5.3: Overview on activity discovery approach

Step 1: Group emails by sender: This step regroups emails by their senders, which will be useful to consider employees' writing style when discovering their frequent activities. We suppose that one employee, who is frequently implied in the execution of one or multiple activities, is likely to use close wordings to express the same frequent activity. Such close wordings would bring a kind of regularity to the coherent expressions that he/she uses for

expressing activities' components. This regularity will be reflected by the set of patterns shared by coherent expressions, which are used to express the same activity.
In other hand, this step would be useful to minimize the variance degree of emails' writing styles. It reduces the degree of noise per employee. Additionally, it could potentially enhance the interpretability and the coherency of the obtained patterns while decreasing the appearance of non-relevant combinations of words.
Step 2: Discover frequent activities per employee: This step generates a list of frequently discussed activities in emails. It performs three elementary sub-steps:
`Sub-step 2.1: Learning frequent patterns of concepts:` It analyzes per employee emails' outbox to learn his/her frequent patterns of concepts forming coherent expressions. Such goal requires matching synonymous words. We rely then on the integration of a dictionary of synonyms (e.g. WordNet[2] for english words).
`Sub-Step 2.2: Classifying frequent patterns of concepts:` This sub-step aims to obtain potential activity names and business information. Therefore, it classifies frequent patterns into action pattern (of action expressions) if they contain at least one verb. Otherwise, they are considered as non-action patterns (reflecting non action expressions). Patterns composed only of verbs are excluded because coherent expressions would not be only formed by verbs. At this level, each obtained action pattern is formed by at least one non-verb concept that may refer to: (1) useful information contributing in specifying activity names, or (2) business information.
`Sub-Step 2.3: Group patterns into activities:` It aims to regroup action and non-action patterns into activities. An activity is associated to multiple action patterns since one employee can express differently activities using words sharing the same context (not necessary synonyms), e.g.; *"I sold 50 mw's"* expresses the activity of selling electricity power whereas the word *'power'* was not explicitly mentioned. Nevertheless, it could be deduced from the mentioned power quantity *"50 mw's"* knowing that *"mw"* refers to the power energy unit *"megawatt"*.
To ensure such regrouping, we suppose that patterns of the same activity must share:

(i) Similar action: Two patterns have similar actions if their verb actions have synonymy relation (e.g. *'check deal'* and *'ensure deal'*) or rewording relation (e.g. *'interview candidate'* and *'conduct interview'*); AND

(ii) Similar business context: The business context of each pattern is inferred from the set of non action patterns that highly coexist with it in emails;

Step 3: Group similar activities of different employees: Different employees collaborate in the execution of one BP instance, that's why it is important to group similar activities executed by different employees. Each activity at this level is characterized by: (i) a set of action patterns sharing the same verb concepts, (ii) one verb concepts used to group such action patterns in sub-step 2.3, and (iii) non-action patterns that represent activity business information. In this step, we proceed as in sub-step 2.3 by considering that similar activities must share similar action and business context.

[2] https://wordnet.princeton.edu/

In what follows, we will detail only the key steps of our proposal, namely sub-step 2.1 (*learning frequent patterns of concepts*) and sub-step 2.3 (*Group patterns into activities*).

5.3.2 Learning frequent patterns of concepts

This sub-step aims to learn/discover the frequent patterns of concepts shared by the overall of emails sent by the same employee. Existing algorithms for patterns discovery from textual documents are mainly of two types:

(i) Words coexistence based algorithms: They simply adapt FIM (Frequent Item Set Mining) techniques in the context of textual data [61, 6, 39]. FIM techniques have been introduced to find the set of frequent itemsets from database of transactions (e.g. A priori algorithm [6], FP-Growth [38], etc). During the execution, FIM techniques create patterns with items that coexist simultaneously without paying attention to their orders of appearance or their relative positions in the transactions. The selection of important patterns is generally based either on their frequency of appearance or on the degree of correlation of their elements (which is the case of association rules). Nevertheless, in the context of textual data, relying solely on the coexistence of words is likely to generate insignificant patterns (noise) in the case of highly dispersed ones (as previously explained at the level of Section 4.4.1);

(ii) Multiple Words Expressions (MWE) discovery techniques: These techniques gather words on a textual document while paying attention to the relationships between words so that their semantic meanings can be preserved. Based on recent surveys established in this context [62, 63, 72], Sequential Patterns Mining (SPM) techniques present the state of the art today (e.g. n-gram generators, GSP, Spade, PrefixSpan [73], Spam, etc). During the construction of patterns, SPM techniques analyze each textual document separately while relying on the sequence of words to form patterns. More precisely, they generate for each textual document the overall combinations of words that would correspond to potential patterns. Then, they rely on a set of selection criteria (e.g. frequency, correlation degree/conditional probability, etc) to keep the most important ones. However, these techniques are likely to escape the detection of important knowledge. In fact, in the context of textual data; sets of words expressing the same idea do not necessarily appear successively or respect the same sequential order of appearance. Moreover, they are not necessarily expressed using identical words; this is mainly due to two properties related to words: (1) the polysemy property, which is related to multiple meanings of the same word, and (2) the synonymy property where several words have the same meaning.

It is important to note that these techniques mostly require precising the size of patterns to be discovered in advance, so that they could mark the end of patterns growth during their construction. This is limited due to the granularity expression variance, at quantity level, of BP element in emails, which does not allow the size of patterns to be commonly defined in advance for all BP elements.

Based on these limitations, we introduce our own mechanism for discovering patterns of concepts from emails. These patterns differ from the traditional ones generated by the previously cited studies where each one is viewed as a simple combination of words of a predefined size. As previously defined, each pattern of concept allows capturing, for each concept forming it, a set of words having mutual synonymy relations. In this way, we enable the discovery of common expressions between emails even if their words are differently expressed. During the discovery mechanism of patterns of concepts from emails, we do not impose constraints concerning: their size, their word sequencing order or succession. We only impose low dispersion constraints concerning distances separating the position of appearance of each successive words forming each one of them. To this end, relying on analysing each email aside to generate all the possible combination of words forming patterns would be limited. For instance, this requires the definition of pattern size to mark the end of patterns' growth during their generation. Consequently, we alternatively rely on correlating each pair of emails. The goal is to find their intersections in terms of low dispersed patterns of concepts that would refer to the common coherent expressions between them. Hence, the granularity level (i.e. size) of the discovered patterns is to be defined according to the one granted by BP actors (i.e. according to the size of the commonly expressions they frequently write).

We describe in Algorithm 1 the main operations performed in the context of our mechanism for discovering patterns of concepts from emails. These operations are as follows:

(a) Correlating each email pair (line 4);

(b) Updating dictionary of patterns of concepts (lines 5 & 6) after each correlation to capture the trace of: (i) the email IDs where they appear, and (ii) the synonyms that make up their concepts;

(c) Filtering patterns of concepts by frequency (line 7) after defining a threshold of frequency;

In what follows, we will focus on explaining the operations (a) and (b), as operation (c) is trivial.

Algorithm 1 Discover Frequent Patterns Per Employee

1: **procedure** DiscoverFrequentActivitiesPerEmployee($employeeEmails, Th_d, Th_{PatFreq}$)
2: $dic\_PatternOfConcepts = \{\}$, $dic\_Affect = \{\}$, $dic\_ids = \{\}$
3: **for** (em_1, em_2) in $employeeEmails.Pairs()$ **do**
4: $PatternsOfConcepts =$ **Correlation**(em_1, em_2, Th_d) ▷ See Section 5.3.2.1
5: $dic\_PatternOfConcepts, dic\_ids, dic\_Aff =$ **UpdateDic**$(dic\_PatternOfConcepts, dic\_ids,$
6: $dic\_Aff, PatternsOfConcepts, em_1.ID, em_2.ID)$ ▷ See Section 5.3.2.2
7: $dic\_PatternOfConcepts =$ **FilterByFrequency**$(dic\_PatternOfConcepts, dic\_ids, Th_{PatFreq})$
8: **return** $dic\_PatternOfConcepts$

5.3.2.1 Correlating emails' pairs

This operation disposes as input two preprocessed emails. The goal is to find their intersection in terms of patterns of concepts while: (i) verifying the low dispersion criterion of coherent expressions, and (ii) tolerating the existence of synonyms. Given that two emails can share multiple activity components and given that one activity component corresponds to one pattern, our solution for this operation must allow obtaining multiple patterns per correlation.

Towards this goal, the proposed solution first calculates the intersection of the two emails in terms of words sharing similar meanings (to tolerate the use of different words having synonymy relations). Then, it reduces emails to the words belonging to this intersection to search the low dispersed sub-patterns that they are in common between them. More precisely, for each email, it segments it into low dispersed sub-patterns independently of the other email. This is ensured by using a distance position threshold and analysing each pair of successive words in the email to check if they are low dispersed. Then, it calculates the intra-email pairwise intersection between these sub-patterns to infer the ones in common between the two emails. Finally, it uses the obtained sub-patterns to build the patterns of concepts that are in common between the two emails. In what follows, we further detail each of these three stages: find intersection between the two emails, find low dispersed sub-patterns, and build in common patterns of concepts. We additionally rely on: (i) the pseudo-code in Algorithm 2 to resume the main algorithmic functions to be used in each stage, and (ii) example of results obtained from each stage which are related to the emails em_A and em_B.

Algorithm 2 Correlation of two emails

Input: $em_1, em_2, Th_d, dic\_PatternOfConcepts, dict\_synonymy$
Output: $dic\_PatternOfConcepts$
1: MI = **FindMultiMeaningIntersection**$(em_1, em_2, dict\_synonymy)$
2: new_em_1, new_em_2 = **Replace**(em_1, em_2, MI)
3: $MIpos\_em_1$ = **FindOrderedPositions**$(newem_1, MI)$
4: $MIpos\_em_2$ = **FindOrderedPositions**$(newem_2, MI)$
5: $LPpos1$ = **SearchLowDispesedPatterns**$(newe1, Th_d, MCpos\_em_1)$
6: $LPpos2$ = **SearchLowDispersedPatterns**$(newe2, Th_d, MIpos\_em_2)$
7: $intersections$ = **FindPatternIntersection**$(LPpos1,\ LPpos2, Th_d, new\_em_1, new\_em_2)$
8: PC_Words1, PC_Words2 = **RealCorrespondance**$(intersections, em_1, em_2)$
9: $dic\_PatternOfConcepts$ = **UnionPatterns**(PC_Words1, PC_Words2)

Stage 1: Find email pair's intersection (line 2, Algorithm 2): In this stage, we reduce each email to the set of words concatenated with their part of speech category tags. Taking the example of the email em_A, it is reduced to the following set: { *'arrange_v'*, *'preliminary_n'*, *'interview_n'*, *'trader_n'*, *'position_n'*, *'person_n'*, *'availability_n'*, *'contact_n'*, *'define_n'*, *'time_n'*, *'slot_n'*}.
The goal is to find the intersection of the two emails while tolerating the existence of synonyms. Actually, it is insufficient to rely solely on identical words shared by both emails. For instance, the simple intersection of em_A and em_B (in terms of identical words concatenated with their part of speech category tags) results in the following set: $em_A \bigcap em_B$ = { *'preliminary_n'*, *'interview_n'*, *'candidate_n'*, *'time_n'*, *'slot_n'*}. This set does not include the common concepts present in em_A and em_B but that were expressed differently. More precisely, it

does not include: (i) the concept ({*'set'*,*'arrange'*},*'v'*) that reflects organizing an interview, and (ii) the concept ({*'define'*, *'set'*},*'v'*) that reflects fixing a time slot. Consequently, our goal is to search a multi-meaning intersection between the two emails rather than a simple intersection between them. Such multi-meaning intersection is composed of a set of multi-meaning concepts shared by the two emails. We formally define a multi-meaning concept in Definition 5.2. We recall that $\mathcal{T}$ is the set of part of speech categories and $\mathcal{C}$ is the set of concepts.

Definition 5.2. ***(Multi-Meaning Concept)*** *Let $em_1 \in \mathcal{E}$ and $em_2 \in \mathcal{E}$ be two preprocessed emails. A multi-meaning concept in common between em_1 and em_2 is a tuple $MC = (L_{w_{MC}}, t_{MC})$ composed of a list of lemmatized words $L_{w_{MC}} \subset \mathcal{W}$ and a part of speech category tag $t_{MC} \in \mathcal{T}$ such that:*

(i) $\exists\ L_C \subset \mathcal{C}$ of size $m \in \mathbb{N}$ such that:

- *$m \leq |L_{w_{MC}}|$;*
- *$L_C = \{C_1,\ C_2,\ ..,C_m)\}$ where: $\forall i \in [1,m]$, C_i is a concept defined as $C_i = (L_{w_{C_i}}, t_{MC}) \in \mathcal{W}^* \times \mathcal{T}$ (see Definition 4.3);*
- *$\bigcup_{i=1}^{m} L_{w_{C_i}} = L_{w_{MC}}$; AND*
- *$\forall i \in [1,m]$, $\exists j \in [1,m] \mid L_{w_{C_i}} \bigcap L_{w_{C_j}} \neq \emptyset$;*

(ii) $\forall j \in \{1,2\}, \forall w \in L_{w_{MC}}, \exists\ p_i \in \mathbb{N} \mid LEM_P(em_j)[p_i] = w\ \&\ t_{MC} = POSTag_P(w, em_j, p_i)$;

Each multi-meaning concept (Definition 5.2) is the union of a set of concepts where each one (supporting unique meaning) must necessary share at least one synonym with another existing concept. Such condition (i.e. (i) in Definition 5.2) is required in order not to diverge towards totally different meanings within the same multi-meaning concept.

We introduce the notion of multi-meaning concept since it is ambiguous at this level to determine the set of shared concepts supporting unique meanings, especially if one email contains a word that supports two different meanings existing in the other email. Going back to our example, according to WordNet synonyms and the possible meanings appearing in the two emails; (1) *'set'* of em_B belongs to two meanings ($meaning_1$={*'set'*, *'arrange'*}, $meaning_2$={*'set'*, *'define'*}) and (2) the two meanings are not equivalent because *'arrange'* $\notin$ SynFc(*'define'*,*'v'*). Therefore, we cannot replace the word *'set'* by a tag that reflects unique meaning (because we will neglect the other possible meaning and we may prevent the appearance of some patterns). Given that C_1=({*'set'*, *'arrange'*}, 'v') and C_2=({*'set'*, *'define'*},'v') form two concepts where {*'set'*, *'define'*} $\cap$ {*'set'*, *'arrange'*}={*'set'*}$\neq \emptyset$, MC_1=({*'arrange'*, *'set'*, *'define'*},*'v'*) forms a multi-meaning concept in common between the emails em_A and em_B.

Let $\mathcal{MC}$ denote the set of multi-meaning concepts. Based on such notion, we define a multi-meaning intersection as follows (Definition 5.2):

Definition 5.3. ***(Multi-Meaning Intersection)*** *Let $MI \subset \mathcal{MC}$ be a set of multi-meaning concepts and $em_1 \in \mathcal{E}$ and $em_2 \in \mathcal{E}$ be two preprocessed emails. MI is a multi-meaning*

intersection of em_1 *and* $em_2 \Leftrightarrow \forall\ MC = (L_{w_{MC}}, t_{MC}) \in MI,\ \exists\ (p_1, w_1) \in \mathbb{N} \times LEM_P(em_1)$ *and* $(p_2, w_2) \in \mathbb{N} \times LEM_P(em_2)$ *such that:*

(i) $LEM_P(em_1)[p_1] = w_1$ *and* $LEM_P(em_2)[p_2] = w_2$*;*

(ii) $w_1 \in L_{w_{MC}}$ *and* $w_2 \in L_{w_{MC}}$*; AND*

(iii) $t_{MC} = POSTag_P(w_1, em_1, p_1) = POSTag_P(w_2, em_2, p_2)$*;*

A multi-meaning intersection is defined as the set of multi-meaning concepts shared by two emails. If we calculate the multi-meaning intersection between em_A and em_B, we obtain five multi-meaning concepts: MC_1=({*'arrange'*, *'set'*, *'define'*},*'v'*), MC_2=({*'interview'*},*'n'*), MC_3= ({*'preliminary'*},'n'), MC_4=({*'time'*},*'n'*), MC_5 = ({*'slot'*}, *'n'*).

Stage 2: Find Low dispersed sub-patterns (lines 3 $\rightarrow$ 8, Algorithm 2): The goal of this stage is to obtain, from the set of multi-meaning concepts, multiple low dispersed sub-patterns that reflect the common existing coherent expressions in a pair of emails. In the case of our example (em_A and em_B), we must obtain the following set of multiple sub-patterns MSP = $\{SP_1, SP_2, SP_3\}$ = $\{\{MC_1, MC_2\}, \{MC_3, MC_2\}, \{MC_1, MC_4, MC_5\}\}$ that reflects respectively the following ideas: organizing an interview, preliminary interview and defining a time slot.

Towards this goal, this stage performs two main sub-stages. In the first sub-stage (Sub-stage 2.1), it replaces words, in the preprocessed emails, by tags reflecting their multi-meaning concepts (new_em_1 and new_em_2 in line 3 of Algorithm 2). Then, each email is reduced to the list of actual appearance positions of these tags (where elements appear in increasing order) and without removing redundancy ($MCpos\_em_1$ and $MCpos\_em_2$ in lines 4 & 6 of Algorithm 2). After that, by relying on the dispersion constraints characterizing low dispersed sub-patterns (Definition 5.4), it searches the sub-patterns positions that can be deduced from each email ($SPpos1$ and $SPpos2$ in lines 6 & 7 of Algorithm 2). In the second sub-stage (Sub-stage 2.1), it calculates the intersection between each sub-patterns pairs belonging to different emails (*intersections*, line 8 of Algorithm 2). Such intersections contains: (i) the shared sub-patterns by the two emails if they satisfy coherent expressions criteria, and (ii) the actual appearance positions of the elements composing them.

In what follows, we explain how we segment each email into low dispersed sub-patterns of multi-meaning concepts. We additionally illustrate how we infer from their intersections the sub-patterns in common between the two emails. For this purpose, we first formalize the definition of a low dispersed sub-pattern of multi-meaning concepts (Definition 5.4)

Definition 5.4. ***(Low Dispersed Sub-Pattern)*** *Let consider the following notations:*

- $SP = \{MC_1, ..., MC_{n_{SP}}\} \subset \mathcal{MC}$ *be a sub-pattern of multi-meaning concepts of size* $n_{SP} \in \mathbb{N}\backslash\{0, 1\}$*;*
- $em \in \mathcal{E}$ *be a preprocessed email where all multi-meaning concepts of* SP *occurs;*

- $Pos_{SP} = \{pos_1, ..., pos_{n_{SP}}\} \subset \mathbb{N}^*$ *be a set of integers. It refers to one possible combination of the multi-meaning concepts' occurrence positions of the sub-pattern* SP *in the email* em*;*

- $Sort : \mathbb{N}^* \rightarrow \mathbb{N}$ *be the function that sorts a list of integers in an ascending order;*

- $f_{Dispersion} : \mathcal{SP} \times \mathcal{E} \rightarrow IN^*$ *be the function that returns, for a given sub-pattern, a set of distances referring to its dispersion in a given email such that:* $f_{Dispersion}(em, Pos_{SP}) = [\{|Sort(Pos_{SP})[i+1] - Sort(Pos_{SP})[i]|, i \in [1, |SP| - 1]\}]$*;*

- $Th_d \in \mathbb{N}\backslash\{0\}$ *be a distance threshold between two successive words in a preprocessed email;*

The sub-pattern SP *is low dispersed w.r.t the positions of occurrences of its multi-meaning concepts* Pos_{SP} *in the email* $em \in \mathcal{E}$ *if it verifies the following low dispersion constraints of coherent expressions:* $\forall$ *distance* $\in f_{Dispersion}(SP, em)$*,* $distance \leq Th_d$

The dispersion function (i.e. $f_{Dispersion}$ in Definition 5.4) of a sub-pattern SP in an email em reflects the closeness of the occurrence positions of its multi-meaning concepts. It returns a list of integers referring to the distances between the occurrence positions of each pair of successive multi-meaning concepts composing the sub-pattern SP.
Based on such notion of sub-pattern dispersion, we equally define in Definition 5.4 the criterion of low dispersed sub-patterns. This criterion requires that all distances between each pair of successive multi-meaning concepts are lower than a predefined threshold (Th_d). However, it does not impose additional constraints concerning the sequencing or the appearance order of these latter (e.g. gap between concepts is allowed).

To find the low dispersed sub-patterns (lines 7&8, Algorithm 2) in an email, we propose an iterative approach where we scroll through the list of multi-meaning concepts' positions associated to the email. We add each element to the same sub-pattern if and only if: (i) its multi-meaning concept was not previously added, and (ii) the distance that separates it from those preceding it does not exceed the defined threshold (Th_d). Once having a multi-meaning concept position where dispersion constraints of the new sub-pattern is not verified, this latter marks the end of one potential sub-pattern and the beginning of another.

Getting back to our example of the two emails em_A and em_B, let take 3 as a distance threshold value (i.e. Th_d=3). We illustrate in Figure 5.4 the obtained results after performing the two sub-stages of the first stage (i.e. Sub-stage 2.1 & Sub-stage 2.2 in Figure 5.4). For the first sub-stage, we show for each email (i.e. em_A & em_B):

- The list of terms obtained after replacing those belonging to in common multi-meaning concepts by tags (i.e. new_em_A & new_em_B);

- The corresponding positions of occurrence of each term in each list (see the positions colored in blue);

- The set of position pairs of each pair of successive multi-meaning concepts (see the position tuples colored in orange);

- The set of low dispersed patterns obtained in each email (see the terms framed by colored rectangles). For instance, we obtain from em_A, $subPat_1\_em_A$ and $subPat_2\_em_A$ and from em_A. We obtain $subPat_1\_em_B$, $subPat_2\_em_B$ and $subPat_3\_em_B$;

For the second sub-stage (see Sub-stage 2.2 in Figure 5.4), after calculating the intersections of the low dispersed patterns, we return I_{11}, I_{13} and I_{22} as well as their real positions in the two emails after checking these two elements: (i) they verify dispersion constraints, and (ii) their size is greater than 2. For the other intersections (i.e. I_{12}, I_{21} and I_{23}), they are not returned because their sizes are lower than 2 which means that they can not form patterns.

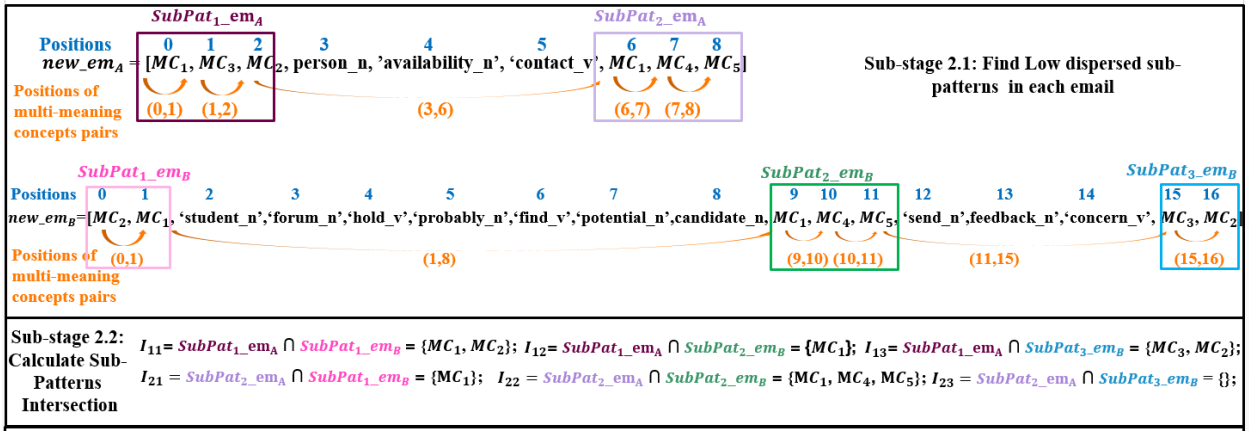


Figure 5.4: Correlating em_A and em_B: Stage 1

Stage 3: Build Patterns of Concepts (lines 9 & 10, Algorithm 2): The goal of this stage is to deduce the shared patterns of concepts (between two emails) from the *intersections* generated from *Stage 2* (line 8, Algorithm 2). To this end, for each sub-pattern, we target to identify the actual meanings that their multi-meaning concepts express, depending on their local context of appearance in both emails. For this purpose, we first retrieve the real correspondences of each multi-meaning concept forming the discovered sub-patterns (which means the actual words related to them in both emails). We illustrate, in Figure 5.5, the real correspondences of the obtained intersections between the sub-patterns of the emails em_A and em_B in the the table shown in Stage 3. More precisely, we resume in the lines $RealCorresp(I_{ij}, em_A)$ and $RealCorresp(I_{ij}, em_B)$ the real correspondences of each intersection I_{ij} in the emails em_A and em_B, e.g. $RealCorresp(I_{11}, em_A) = [new\_em_A[0], new\_em_A[2]] = [$*'arrange_v', 'interview_n'*$]$). Finally, for each set composed of the real correspondences of the same multi-meaning concept, we check if they have reciprocal synonymy relation before merging them in the same concept (see the table in Stag 3 of Figure 5.4, line UnionCorresp). In this way, we identify the unique meanings of the multi-meaning concepts according to their local context and their position of occurrence in the two emails.

Stage 3: Build Patterns of Concepts

I_{ij}	I_{11}	I_{13}	I_{22}
RealCorresp(I_{ij}, em_A)	[new_em_A[0], new_em_A[2]]=['arrange_v', 'interview_n']	[new_em_A[1], new$_em_A$[2]]= ['preliminary_v', 'interview_n']	[new_em_A[6], new_em_A[7], new_em_A[8]]= ['define_v', 'time_n', 'slot_n']
RealCorresp(I_{ij}, em_B)	[new_em_B[1], new$_em_B$[0]]=['set_v', 'interview_n']	[new_em_B[15], new$_em_B$[16]]= ['preliminary_v','interview_n']	[new_em_B[9], new$_em_B$[10], new_em_B[11]]= ['define_v','time_n','slot_n']
UnionCorresp	{{'arrange_v',' set_v'},{'interview_n'}}	{{'preliminary_v'},{'interview_n'}}	{{'set_v', 'define_v},{'time_n'},{'slot_n'}}

Figure 5.5: Correlating em_A and em_B: Stage 2

5.3.2.2 Updating dictionary of patterns of concepts (line 5, Algorithm 1)

At each iteration, after correlating a new pair of emails, we use two dictionaries (i.e. the two variables *Dic_patternConcepts* and *dic_ids* in Algorithm 1) to respectively store and update the obtained set of patterns and the email IDs where they appeared (which is useful to infer their frequency of appearance). For each pattern of this set, if it is similar with another one obtained from another correlation (i.e. of another couple of emails), it will be updated in the case of having new synonyms that were not previously affected to it. Otherwise, this pattern will be affected to the dictionary with a new key. In the overall of updating step, the trace of the appearance (emails IDs) of each detected pattern is preserved (using *dic_ids*) to infer latter its frequency. Supposing at an iteration it_1, our dictionary contains a pattern of concepts PC_1={({$'get'$, $'have'$},$'v'$), ({$'reservation'$},$'n'$)}. Then, in another iteration it_2 ($it_2 > it_1$), we obtain PC_2={({$'get'$, $'take'$},$'v'$), ({$'reservation'$},$'n'$)} among the shared patterns. As the elements of {$'get'$, $'take'$, $'have'$} are reciprocally synonyms, PC_1 will be updated, in iteration it_2, by the new term $'take'$; the dictionary *Dic_patternConcepts* will contain PC_1={({$'get'$, $'have'$, $'take'$},$'v'$), ({$'reservation'$}, $'n'$)}.

5.3.3 Group patterns into activities

The goal here is to regroup patterns into activities after classifying them into action and non-action patterns. We remind that an activity has two main components: (i) activity name, and (ii) business information. Therefore, to discover activities, it is mainly about discovering their components and recognizing those that belong to the same activities. We recall that we have assumed that action patterns would potentially refer to activity names. As for non-action patterns, they would potentially refer to activity business information. We basically propose to regroup action patterns into activity names and then, to identify for each activity name, the set of highly correlated non-action patterns (in terms of coexistence) to form their business information. We assume that a group of action patterns referring to the same activity name have to share: (i) the same main action, and (ii) the same business context.

Given these assumptions, to regroup patterns into activities, we perform a set of operations that we explain according to five main stages (see Figure 5.6). In the first stage (a) (Section 5.3.3.1), we prioritize the regrouping of action patterns by inclusion. We aim from this stage to reduce the potential redundancy of certain word combinations across action patterns. In the second stage (b), we regroup the obtained groups of included patterns

by main action similarity (Section 5.3.3.2). We additionally identify, in the third stage (c) their highly correlated/coexisted non-action patterns that form their business information (Section 5.3.3.3). Afterwards, in the fourth stage (d) we use these business information to regroup action patterns per each main action into activities (Section 5.3.3.4). Finally, we infer the business information of each activity at the level of the fifth stage (e) (Section 5.3.3.5).

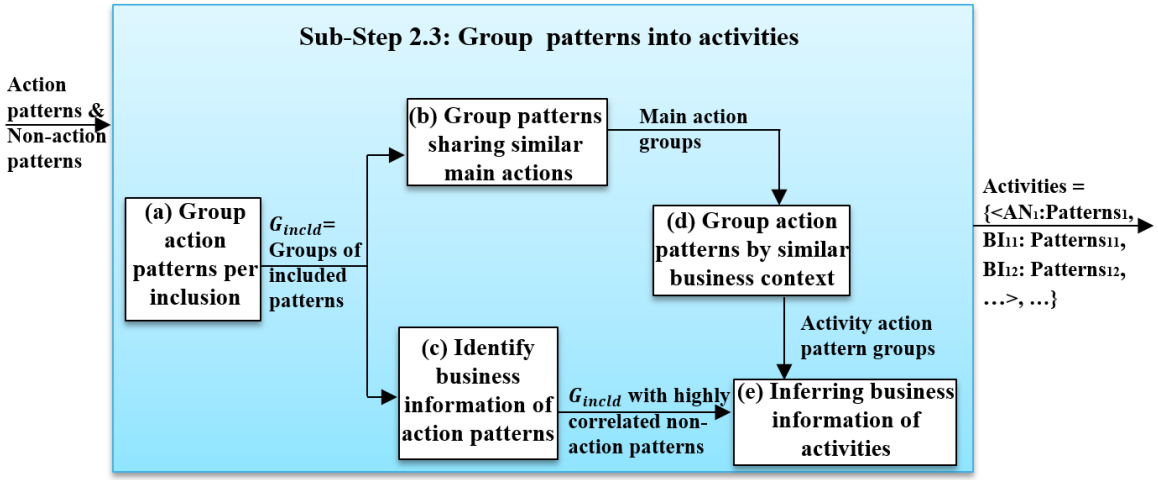


Figure 5.6: Main stages for grouping Patterns into activities

5.3.3.1 Group action patterns per inclusion

Due to the various expression levels of BP elements, namely at quantity level, employees could use combinations of words of various sizes (i.e. number of words) to express the same activity name. One of the main reasons is to specify the expressed activity name by words referring to the corresponding business context or business data. We take the following examples for illustrating such situation:

- Example 1: For requesting an interview arrangement, a manager could ask his/her assistant using these two options: (i) *'could you please arrange an interview with this candidate ?'*, or (ii) *'could you please arrange a phone interview with this candidate ?'* while specifying the interview type (i.e. phone interview). If these two options are frequently used, our solution of patterns discovery will generate two types of patterns PC_{11}=$\{(\{'arrange'\},'v'), (\{'interview'\}, 'n')\}$ and PC_{12}=$\{(\{'arrange'\}, 'v'), (\{'phone'\}, 'n'), (\{'interview'\}, 'n')\}$;

- Example 2: For informing about the extension of one deal, an employee could use these two options: (i) *'I extended the deal'*, or (ii) *'I extended the deal 454057'* while mentioning the deal number. If these two options are frequently used, our solution of patterns discovery will generate two types of patterns PC_{21}=$\{(\{'create'\}, 'v'), (\{'deal'\}, 'n')\}$ and PC_{22}=$\{(\{'create'\}, 'v'),(\{'deal'\},'n'), (\{'numeric'\}, 'n')\}$;

We could notice that PC_{11} is included in PC_{12} and equally, PC_{21} is included in PC_{22}. Additionally, PC_{11} and PC_{21} would logically occur more than PC_{12} and P_{22}, respectively. In fact, they have to appear when PC_{12} and PC_{22} occur in emails. To handle such situation, we start our step of grouping patterns per activity by grouping the included ones. To this end, we identify those of size 2 (i.e. containing two words), which present the minimal possible size. Then, for each one, we identify all the patterns of bigger size containing it. We obtain at this level a set of groups of patterns where each one is centered around a pattern of size 2. Finally, we merge groups that share the same patterns to obtain the final set of per inclusion patterns' groups.

In the following steps, for each group, we only pick the action pattern of minimal size to represent the other patterns included in them. We equally pick the patterns that are not assigned to existing inclusion groups. In other words, we reduce our set of action patterns into such picked ones to reduce information redundancy. It is important to note that in the case of an action pattern that represents an inclusion group, the email IDs where it appears will be equal to the union of the email IDs where the patterns belonging to its group appear.

5.3.3.2 Group patterns sharing similar main actions

This step regroups action patterns expressing similar realization type (that could be deduced from their action verbs). It performs two main operations. In the first one, it regroups similar verb concepts existing in action patterns. In the second one, it assigns each action pattern to the corresponding group of verb concepts.

For performing the first operation, we define a group of verb concept as follows:

Definition 5.5. ***(Verb concept group)*** *Let* $Gr_{concept} \subset \mathcal{C}$. $Gr_{concept}$ *is group of concept if* $\forall(c_1, c_2) \in Gr_{concept} \times Gr_{concept}$, $|c_1 \bigcap c_2|$=min($|c_1|$,$|c_2|$)

In other words, concepts of the same group must have pairwise inclusion so that the number of potential meanings that they represent would not deviate from one meaning.
For performing the second operation, we suppose that an action pattern is to be assigned to a group of verb concepts if it verifies one of the following conditions:

- **Condition 1**: Its verb concept belongs to a verb concept group;
- **Condition 2**: The combination of its verb and one of its noun concepts is the rewording of a verb assigned to a verb concept group. Taking the example of the pattern forming the expression *'make arrangement'*, it is the rewording of the verb *'to arrange'*. As for the patterns forming the expressions *'conduct interview'* or *'make interview'*, they are the rewording of the verb *'to interview'*;

Towards this goal, we start by checking the second condition for all patterns as a first step to assign them to the corresponding verb concept groups. Then, for the remained patterns,

we use the first condition. To check the second condition for each pattern, we retrieve its verb and noun concepts. Then, we recognize the verb origin of the noun concepts. For English words, we use Wordnet to this end as it enables retrieving the derivationally syntactic related terms of any word including its verb root form. Finally, we check if the verb root of one of the pattern's noun concepts belongs to an existing group verb. If so, we additionally check if the corresponding pattern verb has a realization meaning, which means similar to *'make'*, *'do'*, *'perform'*, *'realize'*, *'effectuate'*, etc.

5.3.3.3 Identify business information of action patterns

To identify business information of action patterns, we suppose that activity names and business information of the same activity would highly coexist in the same emails without necessarily appearing close to each other. Consequently, we identify at this level the highly correlated non-action patterns (that present the potential business information) with action patterns in terms of coexistence in emails. In the following, let $\mathcal{PC}_{act}$ denote the set of action patterns and $\mathcal{PC}_{nnact}$ denote the set of non action patterns. We define the coexistence coefficient between an action and non-action patterns as follows (Definition 5.6):

Definition 5.6. ***(Pattern coexistence coefficient function)*** *Let $f_{P_{id}} : \mathcal{PC} \rightarrow \mathcal{ID}^*$ be the function that returns the set of email IDs where a pattern $PC \in \mathcal{PC}$ appears. The pattern coexistence coefficient $f_{P_{coeff}} : \mathcal{PC}_{act} \times \mathcal{PC}_{nnact} \rightarrow [0, 1]$ is the function that measures how much an action pattern $PC_{act} \in \mathcal{PC}_{act}$ coexists with a non action pattern $PC_{nnact} \in \mathcal{PC}_{nnact}$ such that;*

$$f_{P_{coeff}}(PC_{act}, PC_{nnact}) = \frac{|f_{P_{id}}(PC_{act}) \cap f_{P_{id}}(PC_{nnact})|}{min(|f_{P_{id}}(PC_{act})|, |f_{P_{id}}(PC_{nnact})|)}.$$

A non action pattern (PC_{nnact}) is considered highly correlated with an action pattern (PC_{act}) if $f_{P_{coeff}}(PC_{act}, PC_{nnact}) \geq Th_c$ where $Th_c \in]0, 1]$ is a user parameter. Logically, the value of such parameter must not be close to zero to be consistent with our objective of bringing out the *'highly correlated'* non action patterns with an action pattern.

5.3.3.4 Group action patterns by similar business information and deduce activity names

For each main action group (returned by *Step 2*), we regroup its action patterns by similar business context (that is characterized by their highly correlated business information). Each obtained action pattern group will represent an activity whose activity name is to be deduced from the most recurrent pattern. In what follows, we first define the pairwise similarity measure between action patterns (Definition 5.7). Then, we define a group of action patterns defining an activity. Finally, we detail our grouping mechanism to obtain such groups.

Definition 5.7. ***(Action Patterns Pairwise Similarity)*** *Let consider the following notations:*

- $PC_{1_{act}}$ *and* $PC_{2_{act}}$ *be two action patterns* $\in \mathcal{PC}_{act}$*;*
- $S_i \subset \mathcal{PC}_{nnact}$ *be the highly correlated set of non action patterns with* PC_i*, where* $i \in [1, 2]$*;*
- $f_{BC} : \mathcal{PC}^*_{nnact} \rightarrow \mathcal{W}$ *be the business context function that returns from a set of non action patterns the set of business context words forming them and that are different from numeric and named entities' tags:*
- $f_{W_{id}} : \mathcal{W} \rightarrow \mathcal{ID}$ *be the function that returns for a word* ($\in \mathcal{W}$)*, the set of email IDs where it appears;*
- $f_{P_{id}} : \mathcal{PC} \rightarrow \mathcal{ID}^*$ *be the function that returns the set of email IDs where a pattern* $PC \in \mathcal{PC}$ *appears*
- $f_{P_{coexist}} : \mathcal{PC}_{act} \times \mathcal{W}^* \rightarrow \mathbb{R} \cap [0, 1]$ *such that* $f_{P_{coexist}}(PC_{act}, B) = \frac{|\bigcup_{b \in B} f_{W_{id}}(b) \cap f_{P_{id}}(PC_{act})|}{|f_{P_{id}}(PC_{act})|}$ *be the function that measures how much an action pattern* (PC_{act}) *coexists with a set of business context words* ($B \subset \mathcal{W}$)*;*

The pairwise similarity between two patterns $PC_{1_{act}}$ *and* $PC_{2_{act}}$ *is defined as a tuple* $Sim_{pairwise} = (Sim_P(PC_{1_{act}}, PC_{2_{act}}), B_{inter})$ *where:*

- $B_{inter} = f_{BC}(S_1) \cap f_{BC}(S_2)$*;*
- $Sim_P : \mathcal{PC}_{act} \times \mathcal{PC}_{act} \rightarrow [0, 1]$ *is the similarity measure function defined as follows:*

$$Sim_P(PC_{1_{act}}, PC_{2_{act}}) = min(f_{P_{coexist}}(PC_{1_{act}}, B_{inter}), f_{P_{coexist}}(PC_{2_{act}}, B_{inter}));$$

The pairwise similarity of two action patterns is then defined according to two elements: (i) the set of business context words that are shared between them (B_{inter}), and (ii) a similarity measure ($Sim_P(PC_{1_{act}}, PC_{2_{act}})$) that reflects the degree of coexistence of such set of business context words with both action patterns. By picking the minimum value of the coexistence coefficients, two action patterns will have a high similarity measure if and only if both of them have high coexistence coefficient with the set of business context words that they share. Based on such notion of pairwise similarity, we define a group of action patterns forming an activity as follows (Definition 5.8):

Definition 5.8. ***(Activity pattern actions Group)***
Let consider the following notations:

- $f_{BC} : \mathcal{PC}_{act} \rightarrow \mathcal{W}$ *be the function that returns for each action pattern* $PC \in \mathcal{PC}_{act}$*, the set of its related business context words;*
- $f_{P_{coexist}} : \mathcal{PC}_{act} \times \mathcal{W}^* \rightarrow \mathbb{R} \cap [0, 1]$ *be the function that measures how much an action pattern coexists with a set of business context words as defined in Definition 5.7;*
- Th_s *and* Th'_s *be two user parameters such that* $Th_s \in]0, 1]$ *and* $Th'_s \in]0, 1]$

$G_{Act} \subset \mathcal{PC}_{act}$ is considered as a group of action patterns forming an activity $Act \in \mathcal{A}$ such that: (1) $B_{G_{Act}} = \bigcap_{PC_{act} \in G_{Act}} f_{BC}(PC_{act})$, and (2) $\forall PC_{1_{act}}, PC_{2_{act}} \in G_{Act} \times G_{Act}$ where $PC_{1_{act}} \neq PC_{2_{act}}$:

(i) $Sim_P(PC_{1_{act}}, PC_{2_{act}}) \geq Th_s$ (see the expression of Sim_P in Definition 5.7);

(ii) $f_{P_{coexist}}(PC_{1_{act}}, B_{G_{act}}) \geq Th'_s$; AND

(iii) $f_{P_{coexist}}(PC_{2_{act}}, B_{G_{act}}) \geq Th'_s$;

A group of action patterns is then defined as a set of action patterns such that ***every pair has a direct similarity relation*** according to the same degree of pairwise similarity (i.e. (i), Definition 5.8) and to the same business context features (i.e. (ii) and (iii), Definition 5.8). Such degree is actually defined by two user parameters (i.e. Th_s and Th'_s) that must not be logically close to zero to guarantee that patterns have considerable degree of business context similarity. This is useful for building groups of action patterns while potentially maximizing the consistency of those belonging to the same group in terms of:

- Business context: This means that all action patterns have to be used in at least one business context in common;
- The activity that they refer to: this means that ideally, action patterns of the same group have to refer to one activity and do not diverge to referring to multiple ones.

Given such quality of pattern groups that we target to obtain, we have to adopt a grouping mechanism that: (1) fits its defined criteria in Definition 5.8, and (2) meets the feasibility of its integration in our context. For the most common clustering algorithms (e.g. kmeans [40], hierarchical clustering [68, 12], Dbscan [34], Hdbscan [16], etc), we have highlighted limitations at the level of the previously cites aspects:

- ***Fitting the defined grouping criteria:*** This mainly concerns clustering algorithms whose formal definitions of the finally obtained clusters does not fit the grouping quality that we target to obtain. Given the example of the Dbscan algorithm [34] (which is also valid for the Hdbscan algorithm [16]), according to the corresponding formal definitions [16], a cluster is defined as a subset of data such that every pair is density-connected. A pair of data is density connected if its members are directly or transitively $\epsilon - reachable$. This means that: (i) their dissimilarity is less than a distance ϵ (i.e. directly $\epsilon - reachable$), or (ii) they have a number of in-common data having with them a dissimilarity less than a distance ϵ (i.e. transitively $\epsilon - reachable$). Such notion of transitivity is actually contradictory with our grouping criteria. In fact, we require direct similarity relations between every pattern pair in the same pattern group forming an activity. As we previously explained, this is useful to ensure its consistency in terms of one potential business context and one potential activity that it would refer to (i.e. to not diverge to referring to multiple ones if we adopt a pairwise similarity relation per transitivity);

- ***Feasibility of integration in our context:*** This concerns clustering algorithms requiring the a priori definition of a set of parameter values that are directly or indirectly related to the number of clusters to be discovered. These parameters could not be defined in an absolute way as they strongly depend on the data to be partitioned. In other words, we could not estimate the limits of their values in advance. Giving the example of the number of clusters (as in the case of Kmeans), it is a parameter that strongly depends on the set of data to be partitioned (so it could not be generalized for all kinds of partitions). As for the level cut parameter required for a hierarchical algorithm, it indirectly intervenes in precising the number of clusters. Nevertheless, it strongly depends on the heights of the nodes that form the hierarchical tree illustrating the arrangement of the merged clusters at each iteration (because it is to be selected from these heights). The height of these nodes increases with the level of the merger as each one is proportional to the sum of the inter-group dissimilarities between the merged child groups. This actually confirms that the value of such parameter strongly depends on the set of the data to be partitioned and could not be generalized for all cases.
  The most adopted solution to handle such situation is to run such kind of algorithms multiple times on the same data to be partitioned to find the parameter values ensuring the optimal number of clusters. This is actually limited as in our case, it is to be applied on each group patterns sharing similar main action to be portioned into activities. Knowing that we logically have several main action groups, applying such solution would be time consuming (e.g. if we dispose 10 main action groups and for each group, we vary a parameter according to a number d of possible values, we have to run the same algorithm $10 \times d$ times). Additionally, it is not feasible to estimate, for each main action group, the number of activities or at least it variation interval. This prevents optimizing the total number of running the same clustering algorithm for the total main action groups. Furthermore, the optimization criteria used by these algorithms to find the optimal number of clusters or to separate them during the grouping mechanism (e.g. maximising inner-group similarity and minimizing intra-group similarity) does not fit our grouping criteria (i.e. the criteria (i), (ii), (iii) in Definition 5.8 that would guarantee consistency in terms of the features that member groups share).

In view of these limitations, we introduce our own grouping mechanism that simultaneously integrates these two aspects (Algorithm 3). The additional requirement to be integrated in this mechanism concerns the criteria (ii) and (iii) in Definition 5.8. In fact, with such membership criteria of the pattern groups, we do not only consider the pairwise similarity measures of action patterns. We equally consider their pairwise intersections in terms of business context words that we retrieve from their highly correlated business information. In other words, the regrouping mechanism that we have to adopt (to regroup action patterns) should not rely solely on quantified pairwise similarities between patterns. It must additionally take into account their pairwise intersections (in terms of features/words having inducing the quantified similarity degrees) to potentially ensure the consistency of each obtained group.

To fulfill the above requirements, the Algorithm 3 takes as input: (i) the set of action patterns ($Patterns$) that we want to cluster, (ii) a similarity matrix (M_{sim}) resuming their

Algorithm 3 Group patterns into activities

Input: $Patterns$, M_{sim},$dicPairs_{InterBC}$, Th_s, Th'_s
Output: dic_{groups}
1: $dic_{affectations} = \{\}$, $dic_{groups} = \{\}$, $indexc = 0$, $N = len(activities)$
2: $PatternPairs = [(index_1, index_2)$ **for** $index_2$ **in** $[index_1 + 1, N]$ **for** $index_1$ **in** $[0, N]$ $if M_{index}[index_1][index_2] \geq Th_s]$
3: $PatternPairs$=**Sort**($PatternPairs$,M_{sim}) ▷ `Sort is a function that sorts patterns pairs w.r.t their similarity coefficient in descending order`
4: **if** $len(PatternPairs) > 0$ **then**
5: $currentPair$=$PatternPairs[0]$
6: $AllSCROLLED$=$False$
7: $scrolledPairs$=$[currentPair]$
8: **while not** $AllSCROLLED$ **do**
9: $index_1$=$currentPair[0]$
10: $index_2$=$currentPair[1]$
11: **if** $index_1 \notin dic_{affectations}.keys()$ **AND** $index_2 \notin dic_{affectations}.keys()$ **then**
12: $indexc = indexc + 1$, $key = str(indexc)$
13: $dic_{groups}[key] = [index_1, index_2]$
14: $dic_{affectations}[index_1] = [key]$, $dic_{affectations}[index_2] = [key]$
15: **else**
16: **if** $index_1 \in dic_{affectations}.keys()$ **then**
17: $index_{Exist} = index_1$, $index_{nnExist} = index_2$
18: **else**
19: $index_{Exist} = index_2$, $index_{nnExist} = index_1$
20: $keys_{exist} = dic_{affectations}[index_{exist}]$
21: **for** $groupkey$ **in** $keys_{exist}$ **do**
22: $group_{exist} = dic_{groups}[groupkey]$
23: $indexPairs$=**pairwise**($group_{exist}$) ▷ `pairwise is a function that forms from a list of elements a list of unique pairs covering all its elements`
24: $B_G = \bigcap_{(i_1,i_2)\in indexPairs} dicPairs_{InterBC}[i_1][i_2])$
25: **if** $f_{coeff}(B_G, Patterns[index_{nnExist}]) \geq Th'_s$ **then**
26: $dic_{affectations}[index_{nnExist}] = groupkey$, $index_{nnExist} = index_2$
27: $allaffectedpairs$=$dic_{affectations}.keys()$
28: $notscrolledPairs$=$[p$ **for** p **in** $PatternPairs$ **if** p **not in** $scrolledpairs]$
29: **if not** $len(notscrolledPairs) == 0$ **then**
30: $found = False$
31: **for** p **in** $notscrolledPairs$ **do**:
32: **if** $p[0]$ **in** $allaffectedpairs$ **OR** $p[1]$ **in** $allaffectedpairs$ **then**:
33: $currentPair = p$ ▷ `Pick the next pair`
34: $scrolledpairs.append(p)$
35: $found$=$True$
36: **break**
37: **if not** $found$ **then**:
38: $currentPair = notscrolledPairs[0]$
39: $scrolledpairs.append(notscrolledPairs[0])$
40: **else**
41: $AllSSCROLLED = True$

pairwise similarity measures, (iii) a dictionary ($dicPairs_{InterBC}$) resuming their pairwise similarity in terms of business context words intersection, and (iv) the value of the threshold (Th_s) defining the minimum similarity value from which two action patterns are to be grouped in the same group. During our regrouping mechanism, a list of activity pairs is first generated and sorted in a decreasing order according to two arguments: (i) similarity measure, and (ii) number of occurrence (lines 2 & 3). Only those that have a high similarity measure (according to a pre-defined $Th_s \in]0, 1]$) are kept. At each iteration, our algorithm performs two main operations:

1. **Operation 1**: It picks a pattern pair from the obtained sorted list in the way that it priories completing the groups that have started to be built. Initially, this operation selects the first pattern pair belonging to our sorted list to be first analysed (line 5);

2. **Operation 2**: It checks a set of assigning criteria to decide if the picked pattern pair will be added to an existing group or if it will initiate a new one. For instance, for each picked pattern pair, if one of its activities was not previously affected to one activity group (line 11), it will be used to initiate a new activity group (lines 2 $\rightarrow$ 14). Otherwise, the activity that was previously affected will be identified as well as its related group (lines 16 $\rightarrow$ 20). Then, the non-previously affected activity will be assigned to the same group if it verifies these two conditions (lines 21 $\rightarrow$ 26):

   (i) It is strongly similar with all its members (according to the Th_s threshold);

   (ii) It has a high coexistence coefficient with the common business context words of all its member (lines 24 & 25) (according to the same Th_s threshold);

At each iteration, for picking the next pattern pair to be analysed (Operation 1), the algorithm retrieves the previously assigned patterns to existing groups (line 27). Then, it scans the not analysed pattern pairs (sorted in descending order) to find the first pair containing a pattern that was previously assigned (lines 28 $\rightarrow$ 36). This pair will present the next one to be analysed by the algorithm. Otherwise (In case of not finding a pair containing a previously assigned pattern), the algorithm selects the first pattern pair from the sorted list of the not analysed ones ($notscrolledPairs$, lines 38 & 39).

Finally, to improve the obtained groups, this step removes redundant ones and merges those having inclusion relationships. Additionally, it assigns patterns that would belong to more than one activity to the most frequent one. For the patterns that remain not assigned after analysing all pattern pairs present in the sorted list at the level of line 3, the algorithm assigns each one of them to an independent group that only contains it.

Our operation sorting the list of activity pairs ensures that the higher the similarity measure of an activity pair, the higher priority it has to initiate activity groups and to be analysed. Additionally, it ensures that our algorithm results would not vary from one execution to another. As for our operation that picks next patterns pair to be analysed at each iteration (Operation 1), it would minimize the possibility of having subdivided groups. For instance, let suppose that, at the level of line 3 of our algorithm, we dispose the following sorted list of patterns pairs: $PatternPairs$=[(A, B), (H, C), (D, E), (B, H), (A, C), (B, C), (A, H), ...]. At the first iteration, the algorithm picks the first pattern pair (A, B). At the second iteration, if it picks the second pair as sorted in the list (i.e. without applying Operation 1), we obtain the pair (H, C). The algorithm will find at the level of Operation 2 that such pair do not verify our assigning conditions to an existing group. That is why, (H, C) will be used to initiate a new group. However, by looking to the next pairs in our $PatternPairs$ list, we could notice that A, B, H and C are likely to belong to the same group. In fact, we additionally dispose the pattern pairs (B, H), (A, C), (B, C) and (A, H) that indicate that these four patterns are pairwise similar. By using Operation 1 at the level of the second iteration, the algorithm will prioritise the pattern pair (B, H), then, it will pick (H, C) in the next iteration. In this way, these pattern pairs would contribute in extending the group initiated by (A, B) at the first iteration rather than initiating new ones.

5.3.3.5 Inferring business information of activities

We keep here the non-action patterns that are highly correlated with all the action patterns forming each activity. To this end, we use the same coexistence function as defined in Definition 5.6 to find the highly coexisted non-action patterns with each activity. Finally, we divide them into: (i) business data patterns; they represent those that contain at least one named entity tag (e.g. numeric, personname, locname), and (ii) business context patterns for those that do not contain named entity tags but give indications concerning business context activities' execution.

5.4 Evaluation

This section evaluates the unsupervised learning solution for activity discovery on real emails belonging to Enron dataset. For ensuring the preprocessing phase, we use a set of natural language processing (NLP) libraries, e.g. *re*[3] for detecting values of numeric types (after injecting regular expressions patterns: RegEx) and *Spacy*[4] to detect named entities, to lemmatize words and to detect their subtrees and their syntactic, part of speech and dependency tags.

We recall that this solution of activity discovery is mainly based on discovering frequent patterns of words to be grouped into activities. It depends on the following parameters:

- $Th_{PatFreq}$ (Algorithm 1): A pattern frequency threshold defining the minimum number of emails containing a pattern to be considered as a frequent pattern;
- Th_d (Definition 5.4): The dispersion constraint threshold that defines the maximum distance between two successive words when a pattern occurs in emails (Definition 5.4);
- Th_{Freq}: An activity frequency threshold defining the minimum number of emails containing an activity to be considered as a frequent activity. It is important to note that activity frequency is different from the number of activity occurrences. In fact, with the number of activity occurrences, one email can contribute in more than one occurrence of the same activity (if it appears multiple times);
- Th_c (Definition 5.6): A coexistence coefficient threshold that defines from which value a business information pattern is considered highly correlated to an action pattern;
- Th_s and Th'_s (Definition 5.8): Two similarity thresholds that define from which value patterns belonging to the same group (representing an activity) are similar. For simplicity, we consider Th'_s=Th_s in this evaluation section;

In the following, we first present our methodology for constructing the evaluation dataset (Section 5.4.1). We additionally describe the obtained one and how we use it to define our

[3]https://docs.python.org/3/library/re.html
[4]https://spacy.io/

experiment road-map for evaluating: (i) activity discovery final results and (ii) its two key steps related to learning frequent patterns and grouping patterns into activities. Then, we present the results of these experiments (Section 5.4.2, Section 5.4.3 and Section 5.4.4) where we discuss the effect of the above parameters on the activity results' significance and recall. We additionally justify some parameter values' choices. Finally, we show the effectiveness of the activity discovery solution through examples of results, when it is applied on a sample of Orange emails, i.e. belonging to another organism than Enron (Section 5.4.6).

5.4.1 Dataset consruction & Experiment road-map

To construct the evaluation dataset of the activity learning part, we have to select emails in the way that the discovered activities are to be considered for building an event log representing the ultimate output of the event log generation phase. As we have no prior knowledge concerning Enron BP, we must collect emails while ensuring that the overall dataset is likely to report potential BP fragments. We recall that we suppose that BP fragments are performed by a group of actors that frequently interchange emails. Additionally, Enron database provides emails of a sample of Enron employees but without providing indications on their organization structure. Therefore, we adopt a strategy to collect a sample of emails while ensuring that they potentially belong to actor groups. To this end, we first select four employees having different organizational and business roles. Then, we use their emails for discovering activities by applying our approach explained in Section 5.3. Afterwards, we enrich this set of employees by other employees belonging to their contact network such that: (i) they are involved in activity related emails, and (ii) the trace of their sent emails is present in Enron dataset. We finally obtain nine employees and we select all emails sent by them. We use the overall emails for discovering the final set of activities and their associated patterns.

Table 5.2 summarizes the features of our evaluation dataset for activity discovery. For each

Table 5.2: Evaluation Dataset

Ep	***OR***	***BR***	$\mathbf{N_{em}}$
$\mathbf{E_1}$	Managing Director	Trading	343
E_2	Senior Counsel	Legal	102
$\mathbf{E_3}$	Managing Director	Risk Management	2283
$\mathbf{E_4}$	Assistant		357
$\mathbf{E_5}$	Manager	Logistics	738
E_6	Specialist	Settlements	108
E_7	Specialist	Logistics	100
E_8	Employee	Employee	158
E_9	Specialist	Logistics	80

employee (whose emails are used for generating the final set of activities), his/her business role (BR), organizational role (OR) and the the number of emails (of non empty main body) sent by him (N_{em}) are indicated. Emails of employees whose index is marked in bold (i.e.

E_1, E_3, E_4 and E_5) were used to generate the first list of activities. The employees (i.e. E_2, E_4, E_6, E_7, E_8 and E_9) were obtained after analysing the activity contact network of the first list of employees.

In what follows, we use this dataset to present the results of four experiments. In the first experiment, we apply the first step for learning frequent patterns. Then, we present some statics to show the distribution of significant/non-significant discovered patterns over their number of occurrences in emails. The goal is to study the effect of $Th_{PatFreq}$ parameter on the significance degree of the discovered patterns. We use to this end emails of the four employees E_1, E_3, E_4 and E_5 from Table 5.2. In the second experiment, we evaluate the grouping quality of the discovered patterns into activities using the same set of emails. We discuss how Th_c and Th_s parameters influence such grouping quality. In the third experiment, we assess the recall and the significance degrees of the discovered activities. We study the effect of Th_d and Th_{Freq} parameters on them. For assessing the recall, we only use the emails of the two employees E_1 and E_4. In fact, such experiment requires the annotation of emails in terms of activities that actually exists in them, which is time consuming compared to other types of annotations needed for the other experiments. Finally, we conclude on the adopted parameter values at the level of each experiment. We outline then the discovered activities from the emails of the overall nine employees using these values.

In the following sections/chapters of this report, each discovered pattern of concepts and/or activity name will be reflected by one string where:

- Concepts are separated by white spaces;
- Part of speech category tags of concepts are marked at the end of the string after the character $'T'$ (only in the case of the pattern)
- Concepts' synonyms (without part of speech tags) are separated by '_'. (e.g., 'buy_purchase powerTvn' reflects the pattern of concepts {({$'buy'$, $'purchase'$},$'v'$), ({$'power'$}, $'n'$) };
- Terms forming composed words are separated by the character '0', e.g. $'numeric0mw'$;
- Each activity name takes the label of the group of patterns in inclusion that occurs the most. This label has the following format:

  $$'verb\ concept_1, ..., \{concept_j\}' \text{ where } j < 4$$

  The first word of this label refers to its verb of action (i.e. *verb*). The other words (i.e. $concept_i$, $i \in [1, j]$) refer to the j non-verb concepts that highly coexist with the action verb in the same patterns sorted in descending order. The first concept $concept_1$ is which coexists the most. The other concepts are the ones that come after. Each of them is put between braces ({}). Taking the example of the activity name *create deal*{*numeric*}{*ticket*}, its verb action is $'create'$, the first concept that highly coexists with the verb is $'deal'$ and the other concepts are $'numeric'$ and $'ticket'$;

5.4.2 First Experiment: Distribution of relevant/non-relevant patterns over frequencies of appearance in emails

In this experiment, we are interested in the action patterns discovered from emails of each employee. The goal is to study their degree of relevance w.r.t their frequencies. To this end, we first select all the obtained patterns from emails of the four employees E_1, E_3, E_4 and E_5. We additionally calculate their related frequencies. Then, we annotate each pattern in terms of relevance, i.e. relevant/non-relevant. A pattern is considered relevant if it mostly forms coherent expressions referring to an understandable and significant action when occurring in emails. We mean by 'mostly' here a number of times greater than half of the total number of occurrences of a pattern in the emails. Finally, we generate a bar diagram that illustrates the frequency distribution of relevant/non-relevant patterns. Figure 5.7 shows such diagram. For each value of pattern frequency, it indicates: (i) the total number of patterns having occurred at the same frequency, (ii) the number of relevant patterns (blue color), and (iii) the number of non-relevant patterns (red color) and its related percentage (e.g. 20.8% for pattern frequency equal to 5).

The distribution diagram shows that the number of discovered patterns increases at low

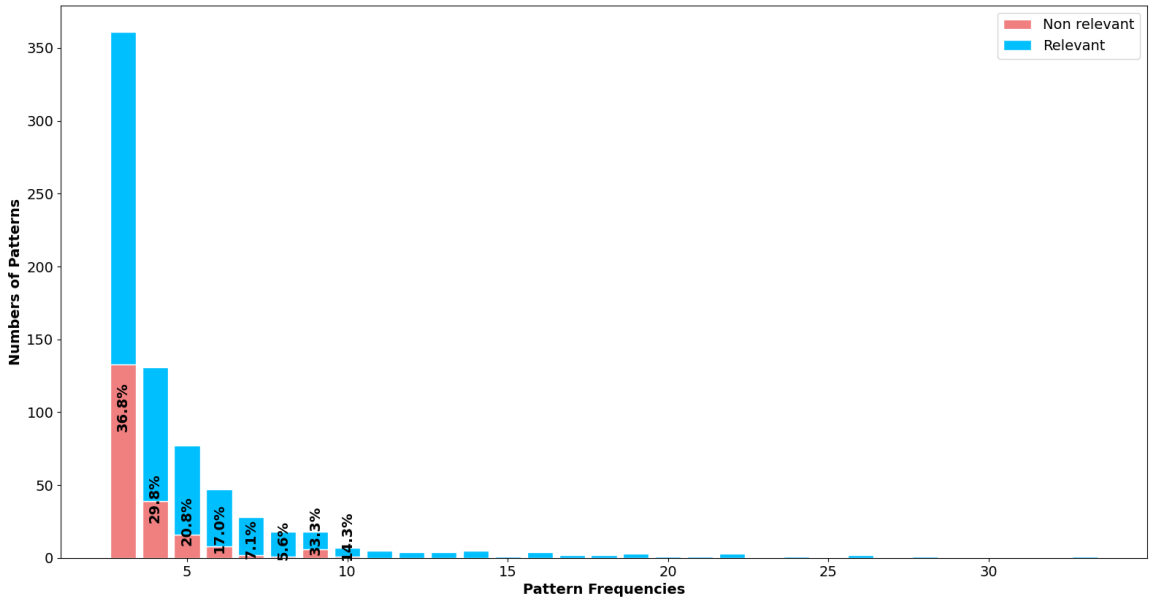


Figure 5.7: Patterns frequencies distribution

frequencies (e.g. 3, 4) as does the probability of being non-relevant (e.g. 36.8% at pattern frequency equal to 3 vs. 0% for pattern frequencies greater than 11). This is logical since patterns appearing at low frequency could correspond to random combinations of words resulted from the intersection of low number of emails. Let take the example of the pattern *attach professorTvn* discovered with the employee E_4 at low frequency equal to 3. In Figure 5.8, we show two emails where it occurred and that have induced its appearance. We can notice that the combination of words *'attached'* and *'Professor'* have appeared closely in the two emails but without referring to a significant expression. As for patterns appearing at high

frequency, they potentially correspond to combinations of words resulted from a high number of intersections/correlations between emails. This increases the probability of referring to a significant meaning. In fact, it is unlikely to have low-dispersed combinations of words that frequently appear without referring to a relevant meaning.

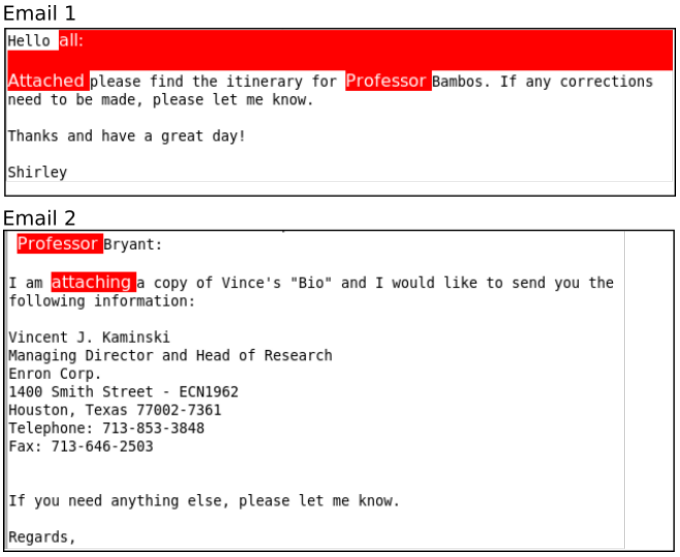


Figure 5.8: Example of Emails resulting in the appearance of the non relevant pattern: *attach professorTvn*

In what follows, we consider a low pattern frequency threshold $Th_{PatFreq} = 3$ to evaluate the performance of our activity discovery approach. With such low value, we logically cover a good part of patterns referring to relevant coherent expressions. Nevertheless, this would probably include non relevant patterns (not referring to coherent expression) as previously discussed.

5.4.3 Second Experiment: Grouping action patterns into activities:

In this experiment, we focus on evaluating the step of grouping action patterns into activities. We recall that this step depends on two main parameters: (i) Th_c that defines the threshold from which a business information pattern is considered highly coexisted with an action pattern, and (ii) Th_s that defines the threshold from which two action patterns are considered similar according to a similarity measure.

The goal in this experiment is to study the effect of these two parameters on the grouping quality of the action patterns. We assess by this, the degree of similarity between the obtained partition of action patterns and the one manually defined. We aim to compare the granularity degree of the two partitions but without being interested in checking the matching between their labels in terms of activities. This granularity degree refers to if one group is divided into multiple sub-groups and inversely, if two groups that must be separated are merged in the discovered ones. Consequently, we use the Adjusted Rand Index (ARI) measure [19] which was introduced to cover such need. It computes a similarity measure between two partitions

by considering all pairs of samples (i.e. patterns in our case). It counts pairs that are assigned in the same or different clusters in the discovered and the manually defined partitions.

In this experiment, we start by selecting all the patterns generated from the emails of the four employees E_1, E_3, E_4 and E_5 at frequency $Th_{patFreq} = 3$. To obtain the manually defined partition, we annotated each pattern w.r.t the activity label to which it contributes to express when appearing in emails. Patterns that were previously identified as non-significant patterns are ignored in this annotation process and are not considered when calculating the ARI measure. However, their effect in terms of generating non-significant activities and erroneous activity occurrences will be assessed in the next experiments. Finally, we generate a graph, that we show in Figure 5.9, to visualize the ARI measure evolution w.r.t the two parameters: Th_c and Th_s.

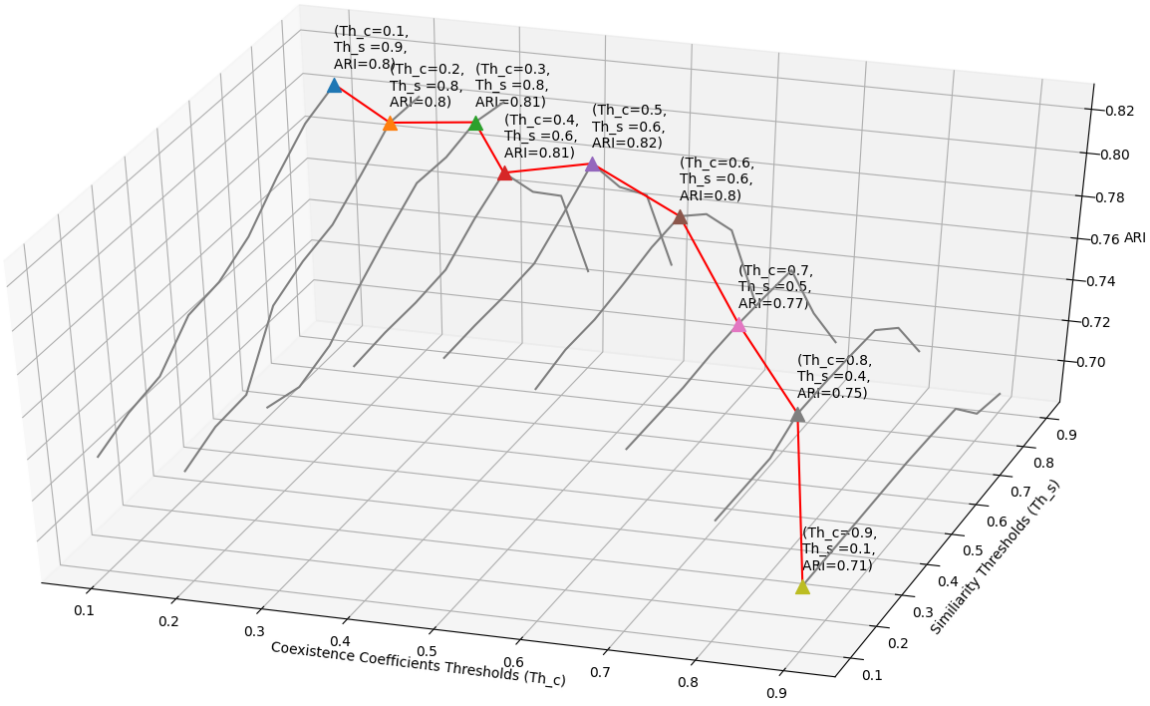


Figure 5.9: ARI evolution w.r.t the parameters Th_c and Th_s

Each gray curve in Figure 5.9 corresponds to a Th_c where $Th_c \in$ [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. It shows the evolution the ARI measrue when varying the value of $Th_s \in$ [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. The maximum ARI value for each curve is marked by a triangular and colored point (e.g. $F1 = 0.81$ for $Th_c = 0.4$ and $Th_s = 0.6$). The overall maximum ARI values are related by a red curve. The grey curves in Figure 5.9 show that for each value of coexistence threshold Th_c, the ARI increases when increasing the similarity threshold Th_s until it reaches a maximum value at a $Th_{s_{max}}$. Then it starts to decrease. The red curve shows that the values of $Th_{s_{max}}$ and Th_c are inversely proportional:

- $Th_{s_{max}}$ tends to 1 for low Th_c values: At low Th_c values, we maximize the number of

non-action patterns representing business information correlated to each action pattern. In this way, the similarity degree between each pair of action patterns tends to increase (as it is deduced from the intersection between their related business information). Consequently, $Th_{s_{max}}$ tends to be close to 1 to restrict the grouping of action patterns.

- $Th_{s_{max}}$ tends to zero for high Th_c values: At high Th_c values, we minimize the number of non-action patterns representing business information correlated to each action pattern. In this way, the similarity degree between each pair of action patterns tends to decrease. Consequently, $Th_{s_{max}}$ tends to be close to 0 to encourage the grouping of action patterns;

- $Th_{s_{max}}$ tends to 0.5 for intermediate values of Th_c: At intermediate values of Th_c, non-action patterns of intermediary level of coexistence with action patterns are considered to calculate patterns' pairwise similarity. This means that we are neither too selective nor too inclusive in terms of the non-action patterns to be considered. Consequently, $Th_{s_{max}}$ tends to be close to 0.5 to be equally of intermediate values;

The overall ARI values vary between a maximum value equal to 0.82 (obtained at $th_c = 0.5$ & $th_s = 0.6$) and a minimum value equal to 0.7 (obtained at $th_c = 0.9$). It is important to note that such minimum value (which is higher than 0.5 in our case) is guaranteed even when increasing/decreasing Th_c and Th_s values. This is due to our steps of grouping action patterns by inclusion and by verb concept. These steps precede the grouping of patterns on the basis of their business information similarity, which would guarantee a stable level of separation between action patterns. In fact, by calculating the ARI value just after grouping action patterns by inclusion, we find a value equal to 0.65. As for the step that groups action patterns by verb concepts, it guarantees that patterns of different verb concepts that have no synonymy or rephrasing relations are initially separated before starting grouping patterns by business context. Nevertheless, it could negatively influence the ARI values if and only if some action patterns having rephrasing relations are not identified and grouped in the same verb concept groups.

To sum up, based on this second experiment, we proof that the values of the parameter th_c and th_s must be logically set close to 0.5. In fact, th_c defines the value of the coexistence coefficient from which a business information is considered highly correlated to a pattern action. Consequently, it could not be logically close to zero. We avoid also values near to 1 to not be too selective and to not discard relevant business information patterns. We adopt therefore in what follows the following parameter values: $th_c = 0.4$ and $th_s = 0.5$ that result in an $ARI = 0.8$.

5.4.4 Third Experiment: Activity Relevance and Recall

In this experiment, we focus on evaluating the performances of the discovered activities in terms of relevance and recall w.r.t two parameters: (i) Th_d: the dispersion constraint threshold defining the maximum distance between two successive words when a pattern occurs in emails,

and (ii) Th_{Feq}: the activity frequency threshold defining the minimum number of emails where an activity appears to be considered as a frequent one.

We evaluate the relevance degree of the discovered activities by calculating the number of the non-significant ones that are obtained in the final results. A non-significant activity corresponds to a group of patterns that, when occurring in emails, we could not infer the activity to which they predominantly refer. We mean by the recall of the discovered results, the capacity of our algorithm to discover the activities that actually exists in a set of emails. In this experiment, we run our activity discovery algorithm on the emails of the four employees E_1, E_3, E_4 and E_5. We consider to this end: (i) $Th_{patFreq} = 3$, (ii) $Th_c = 0.4$, (iii) $Th_s = 0.5$ and (iv) five values of dispersion constraint thresholds (Th_d), which are: 1, 2, 3, 4 and 5. We additionally take into account the business role nature of the trader employee; we define some syntactic function exceptions concerning some trading keywords 'short' and 'long' by setting their functions to verbs (to respect trading terminology because such terms refer to the creation of long/short trading positions). Then, we vary the activity frequency thresholds $Th_{Freq} \in \{5, 6, 7, 8, 9, 10, 11, 12\}$ to study how the relevance and the recall degree evolve. In what follows, we further explain our annotation methodology to calculate the recall of the discovered activities. Then, we comment the obtained results.

5.4.4.1 Emails' annotation in terms of existing activities

This annotation is required to assess the recall degree of the discovered activities. To this end, we select here all the emails sent by two employees (i.e. E_1 and E_4). The first one is a managing trader, he/she has 343 unique emails and his main task is to online trade energy (especially of electricity power). The second employee is an administrative assistant, he/she has 357 unique emails and he/she mainly arranges and coordinates interviews/meetings.

For each email, we manually annotate the list of the discussed activities, the related business data when they exist and the combination of words used by the sender to express them. In such annotation process, we have not considered standard activities (like call, contact or meet persons) or those that can be discovered without the need of analysing the textual content of emails (e.g. the activity *'attach file'* without precising the type of the file can be inferred from the presence of an attached file in the email).

Table 5.3 summarizes the list of the 26 frequent annotated activities that we obtained and that occurred in at least 5 emails. The activity that we find in common between the two employees is marked in blue (i.e. *'Send resume'*). For each employee, the table enumerates the list of the annotated activity names as well as their associated IDs, their occurrence number across emails (Freq) and their related business data (e.g. in the context of trading activities, hourEnding indicates the hour during which trades are conducted, counterpart indicates the seller/buyer that trades with Enron).

Table 5.3: Annotated activities

ID	Activity Name	Freq	Business Data
Managing Trader Activities			
1	Sell power	53	hourEnding
2	Schedule power for trade	29	
3	Buy power	17	
4	Open long trading position	16	zone,
5	Enter deal	13	quantity,
6	Open short trading position	12	price,
7	Replace deal	10	dealNumber,
8	Check deal	7	
9	Cut deal	7	
10	Update resource plan	6	
11	Run plant for production	8	quantity
12	Pay price	12	price,
13	Receive price	7	counterpart
14	Attend meeting	8	
15	*Send resume*	11	
16	Submit schedule	5	counterpart
Administrative Assistant Activities			
17	Conduct interview	45	candidate, position, interviewers
18	Schedule interview	18	date, time
19	Forward resume	14	
20	*Send resume*	13	
21	Schedule meeting	12	date, time
22	Make reservation	11	date, time
23	Send request	9	
24	Reserve conference room	9	roomNumber
25	Arrange interview	9	
26	Send information	8	

5.4.4.2 Experiment results (First Part): Metrics' evolution per Th_{Freq} and Th_d

We show in the figures below the evolution of the recall measure and the number of the non-relevant activities per Th_{Freq} and Th_d. Figure 5.10 illustrates five curves where each one shows the evolution of the recall measure at a $Th_d \in [1, 2, 3, 4, 5]$ when varying $Th_{Freq} \in [5, 6, 7, 8, 9, 10, 11, 12]$ (e.g. blue curve corresponds to $Th_d = 1$). Figure 5.11 equally illustrates five curves where each one shows the distribution of the number of the non-relevant activities per frequency ($N_{Freq} \in [5, 6, 7, 8, 9, 10, 11, 12]$). By using such distribution, we could visualize the level of frequency values where non-relevant activities resided. This helps to understand the effect of Th_{Freq} parameter as it is responsible for discarding such frequency values.

The obtained figures reveal the following observations:

- For the overall dispersion threshold values, the recall decreases while increasing the activity frequency thresholds. This is logical as the more we increase this threshold the more we skip activities with a lower number of occurrences. We notice also that the number of the non-relevant activities decreases until it converges to zero at the level of high activity frequencies. This validates our assumptions where we suppose that: (i) relevant BP activities would be frequently discussed in emails, and (ii) such frequency brings a kind of regularity to the expressions used to express them in emails. This means that the more frequently an activity is discovered, the more likely it is to reflect

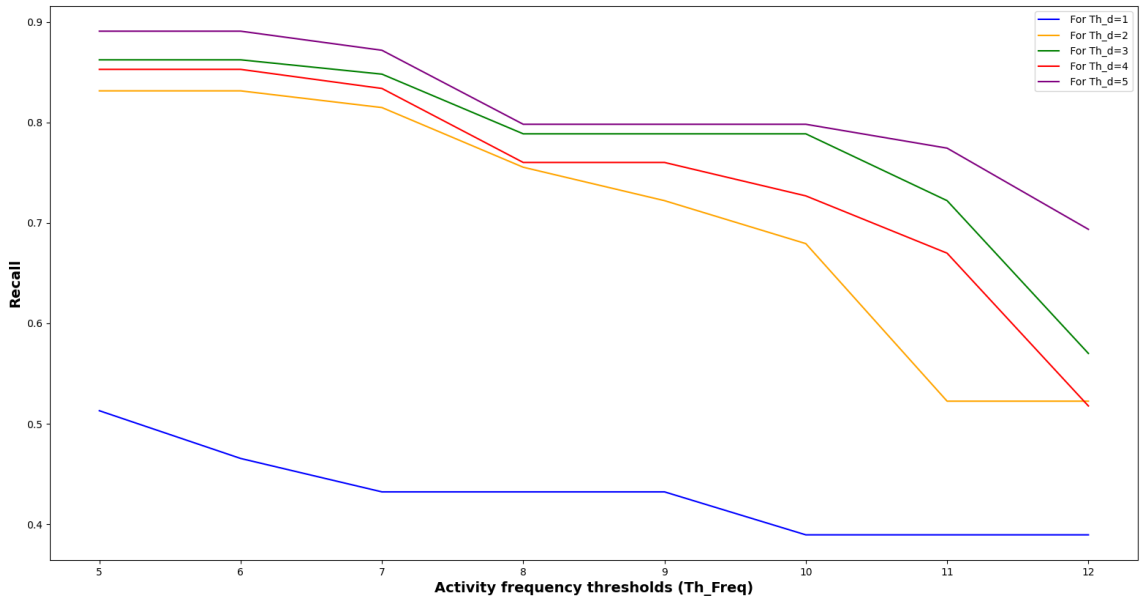


Figure 5.10: Recall evolution w.r.t the parameters Th_d and Th_{Freq}

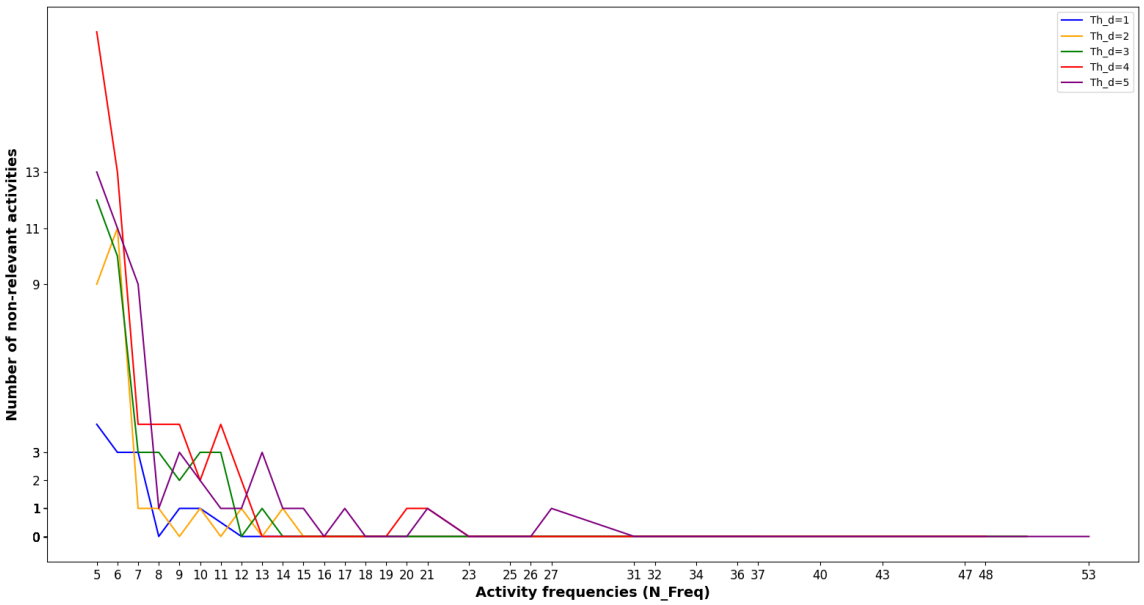


Figure 5.11: Distribution of non-relevant activities w.r.t the parameters Th_d and N_{Freq}

a relevant activity;

- For the minimum possible value of the dispersion threshold (i.e. $Th_d = 1$), the number of the non-relevant activities is minimal even at low number of occurrences ($N_{Freq} \in [5, 6, 7]$). Figure 5.11 shows that four non-relevant activities are obtained for $N_{Freq} = 5$ and $Th_d = 1$. However, the performances in terms of recall are the worst for this dispersion threshold (i.e. $Th_d = 1$); they vary between 0.4 and 0.52 according to Figure 5.10. This means that the algorithm tends to skip the discovery of more than

half of relevant activities in emails. For instance, by considering $Th_d = 1$, we impose that each two successive words of the same activity pattern, must appear successively in emails to be considered in the step that discovers frequent patterns. Nevertheless, this is limited as employees could include other words (e.g. adjectives) between two words forming a coherent expression;

- By increasing the value of the dispersion thresholds ($Th_d > 1$), the recall increases (e.g. it varies between 0.8 and 0.9 for $Th_{Freq} \in [5, 6, 7]$). In fact, with such values of dispersion thresholds, we allow the existence of gap between successive words of the same activity patterns when occurring in emails. This better reflects the reality of expressing activities in emails. However, Figure 5.11 shows that the number of non-relevant activities increases in its turn and it reaches its maximum value for $Th_d = 5$. This is understandable because the words that are highly dispersed do not necessary contribute in forming the same coherent expressions, which induces the appearance of non-significant combinations of words;

To sum up, to make a trade-off between activity recall and relevance, we consider in what follows when discovering activities per employee: (i) a dispersion threshold value $Th_d = 3$, and (ii) an activity frequency threshold $Th_{Freq} = 5$ to maximize the recall of the discovered activities. Then we adopt $Th_{Freq} = 7$ after grouping activities of different employees to discard the non-significant ones appearing at the level $N_{Freq} = 5$ and $N_{Freq} = 6$.

5.4.4.3 Experiment Results (Second Part): Distribution of non-relevant activities per employee:

In this second part of our experiment, we focus on the obtained activities for $Th_d = 3$. We show the distribution of the non-relevant ones that are discovered by our algorithm per employee. Figure 5.12 shows these distributions for the four employees E_1, E_3, E_4 and E_5. We can notice that the most non-relevant activities at the level of low frequencies are coming from E_3 (see the red curve) and then from E_5 (see the orange curve). For E_4 and E_1, the figure shows that the discovered activities are most often relevant for different activity frequency values. Based on our analysis of the related emails, this can be explained by two elements:

- The variety of the job nature of these employees and the purpose for which they send their emails; E_1 and E_4 are likely to have regular activities and most often use their emails in BP context. On the contrary, E_3 can send emails to discuss general topics in relation to Enron strategies (without introducing specific activities), which can increase the appearance of random combination of words.

- The variety of the number of emails analyzed per each employee; The employee E_3 that results in the largest number of non-relevant activities has the greatest number of analysed emails (i.e. 2283). Those that result in a lower number of non-relevant activities have less number of analysed emails (i.e. 343 emails for E_1, 357 emails for

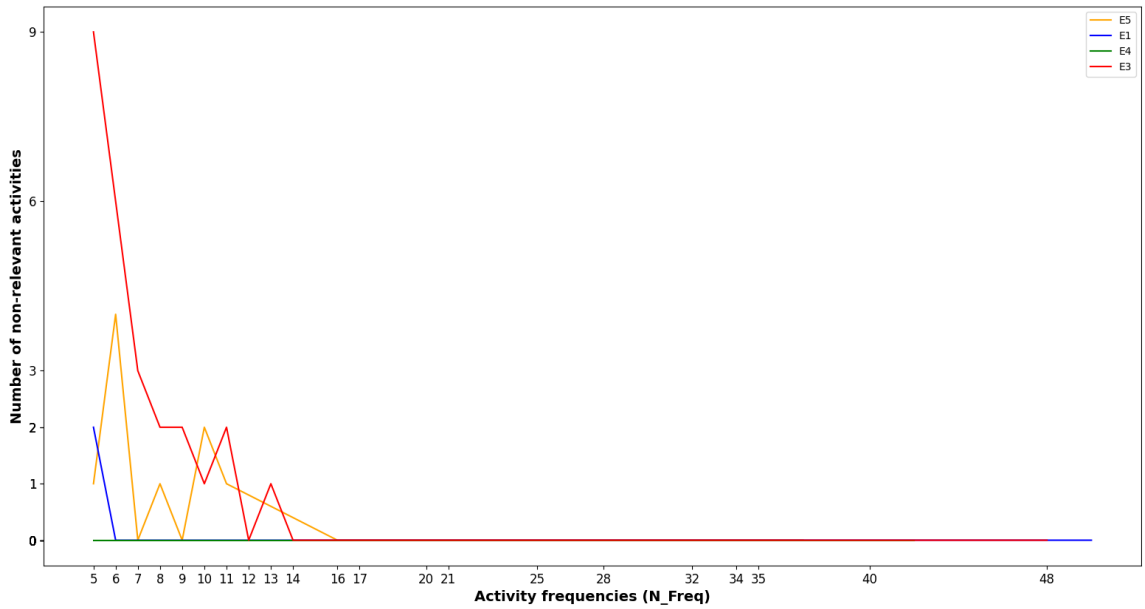


Figure 5.12: Distribution of non-relevant activities per employee

E_4 and 738 for E_5). This can be explained by the fact that as the number of the analysed emails increases, more correlation operations between email pairs are performed. Consequently, more combinations of words could be find in common between emails at different frequency levels. This increases the probability of obtaining frequent patterns that do not refer to significant activities.

5.4.4.4 Experiment Results (Third Part): Recall per activity for the selected parameter values

In this part, we detail the variation of the recall measure per activity for $Th_d = 3$ and $Th_{Freq} = 5$. To this end, we focus on the discovered activities of the two employees E_1 and E_4. We summarize these activities in Table 5.4; we obtained *'attach resume'* as a common activity with both employees (colored in blue in the table) and other non common activities. For each one, the table shows its automatic name generated by our algorithm, the number of emails where our algorithm identifies it (Freq), the ID of its related annotated activity (Id_a), the recall (R) and the precision (Pr) of its detection in emails and finally, its related business data discovered by our solution. For the discovered activities that do not have correspondences in the annotated ones, we only display their discovered names, the number of emails where they appeared and their precision measures. We define the recall of an activity as the ratio of the correct detection to the total number of emails that actually contain it. Finally, we calculate the weighted average of the recall and the precision of the discovered activities within emails. We obtain a value of $R_{AVG} = 0.88$ and $Pr_{AVG} = 0.92$; this reflects that our algorithm is able to identify a good percentage of emails containing relevant activities with good precision.

We calculate also the recall of the overall activities and their business data, which means

Table 5.4: Discovered activities

	Discovered Name	Freq	id_a	R	Pr	BD
Managing Trader	sell_trade numeric0mw{pricenumeric}{hour0end}	50	1	0.88	0.94	hour0end
	schedule locname{zone}{load}	26	2	0.82	0.92	
	buy_purchase numeric0mw{pricenumeric}{hour0end}	17	3	0.94	0.94	locname
	long numeric0mw{hour0end}{orgname0book}	18	4	1	0.88	numeric0mw,
	enter deal{orgname0book}{numeric}	11	5	0.84	1	orgname
	short numeric0mw{he0numeric}	11	6	0.91	1	pricenumeric,
	replace deal_trade{sale}{power}	10	7	1	1	deal
	check_see deal_trade{numeric}	6	8	0.85	1	numeric,
	cut deal	10	9	0.9	0.9	
	reflect resource0plan{numeric0mw}	7	10	1	0.86	
	run oomc0mean{calc}	12	11	0.87	0.91	numeric0mw
	pay cost_price{balance0energy}	11	12	0.83	0.83	pricenumeric
	receive price{numeric0mw}{balance0energy}	13	13	1	0.53	orgname
	attend meeting	7	14	0.87	1	
	attach_send_bind resume{detail_point}	14	15	1	0.78	
	submit schedule	7	16	1	0.71	
	check_break object	6	-	-	0.83	-
	start_begin hour0end{numeric}	5	-	-	0	-
	affect schedule	5	-	-	1	-
Assistant	conduct interview{telephone}{informal}{eb0numeric}	42	17	0.91	0.97	hour_numeric0hour interview, interview personname, orgname
	forward resume	12	19	0.85	1	orgname
	set_arrange_prepare interview{schedule}	10	25	1	0.9	datenumeric
	make reservation{restaurant}	9	22	0.81	1	orgname, datenumeric
	schedule_reschedule meeting	9	12	0.75	1	numerictime, datenumeric
	schedule_reschedule interview{telephone}	14	18	0.77	1	hour_numeric0hour interview, numerictime, datenumeric
	hold_reserve eb0numeric	8	24	0.88	1	eb0numeric, numerictime
	send_attach resume{interview}	10	20	0.77	1	orgname, personname, datenumeric
	send_attach info_information	7	26	0.62	0.71	personname
	coordinate date0time{schedule}{interview}	11	-	-	1	-
	reach personnelpronoun{telephone0number}{availability}	6	-	-	1	-
	invite personnelpronoun	6	-	-	1	-
	attach bio	5	-	-	1	-
Average				0.88	0.92	

the ratio of the correct discovered activities/business data to the total number of the actual activities/business data that must be discovered from emails. We obtained a value of 1 for activities and a value of 0.91 for business data. This indicates that our algorithm was able to discover all the activities and a good percentage of their business data that must be discovered (from the emails of the two employees). For instance, we can notice that all the IDs in Table 5.3 appear in Table 5.4. Additionally, most the business data in Table 5.3 were reflected in Table 5.4 by explicit patterns. Taking the example of the business data *'dealNumber'* (see Table 5.3), it was discovered in the form of *'deal numeric'* (see Table 5.4). As for the traded quantity of electricity power (i.e. *'quantity'* in Table 5.3), it was discovered in the form of *'numeric0mw'* where *'mw'* is the abbreviation of the power unit (i.e. megawatt). Nevertheless, our algorithm does not recognize one business data (i.e. *'position'* in Table 5.3) that concerns interviewing candidates activities. This is understandable as job positions would be generally

expressed in natural language not through numeric values or named entities. Furthermore, we could notice that our algorithm does not differentiate between the two annotated business data: *'candidate'* and the names of the *'interviewers'* (see Table 5.3). They are both reflected by a single business data, i.e. *'interview personname'* in Table 5.4. This is due to the use of the same named entity type (i.e. *'personname'*) when introducing the values of such business data in emails.

The obtained results in Table 5.4 also reveal that our algorithm can generate highly detailed activities that were not considered in the annotation phase. This can be useful when aiming to discover BP fragments with low granularity. For instance, in the context of setting interviews' schedules; we obtain *'coordinate date0time{schedule} {interview}'* that reflects contacting candidates to ask them for their availability.

5.4.5 Summary on the adopted parameters values and the obtained activities for the nine employees

To remind, on the basis of the previous experiments, we adopt the following parameter values:

- $Th_{PatFreq} = 3$ (The pattern frequency threshold)
- $Th_d = 3$ (The dispersion constraint threshold);
- $Th_{1_{Freq}} = 5$ (Activity frequency thresholds per employee) and $Th_{2_{Freq}} = 7$ (Activity frequency thresholds after grouping similar activities of different employees);
- $Th_c = 0.4$ (The coexistence coefficient threshold);
- $Th_s = Th'_s = 0.5$ (The similarity threshold for grouping patterns into activities);

In what follows, we give an overview on the overall activities obtained after: (i) analysing emails of the nine employees (see Table 5.2) used in the unsupervised learning step, and (ii) grouping the similar ones.

Our algorithm generates a final set of activities of size 102. In Table 5.5, we give an overview on the most 30 frequent activities and we provide in this link [5] the complete list. For each activity of an identifier ID, we detail the set of activity names belonging to different employees and that were grouped in terms of similarity to form its related group (see the *Activity Groups* column). We additionally provide the corresponding employees and the activity frequencies in their emails. Given the example of the activity group of $ID = 3$, three activity names of three different employees were grouped in its context:

- *create deal{numeric}{ticket}* that was discovered from the emails of the employee E_8 with a frequency equal to 53;

[5] http://www-inf.it-sudparis.eu/SIMBAD/tools/processDiscoveryFromEmails/

Table 5.5: The most 30 frequent activities discovered from the emails of the nine employees

ID	Activity Groups	N_{Freq}
1	flow deal{gas}{price}, (E_5, 77) flow meter{deal}{numeric}{volume}, (E_7, 27) flow meter{numeric}{deal}{total}, (E_9, 24) flow numeric{meter}, (E_8, 7)	135
2	enter deal{numeric}{ticket}{meter}, (E_8, 64) enter deal{orgname0book}{numeric}, (E_1, 17) enter deal, (E_5, 15) enter numeric{demand0fee}{deal}, (E_6, 12)	108
3	create deal{numeric}{ticket}, (E_8, 53) create deal{numeric}{ticket}{meter_mtr}, (E_5, 41) create deal, (E_9, 11)	105
4	change_convert deal{numeric}{price}{ticket}, (E_5, 49) change numeric{deal}{demand0fee}{volume}, (E_6, 21) vary_change volume, (E_5, 12) change volume, (E_8, 10)	90
5	conduct interview{telephone}{informal}{eb0numeric}, (E_4, 48) bring_contribute interview{arrangement_organization}, (E_3, 23)	71
6	set_put_fix_determine_adjust deal{numeric}, (E_5, 47) set_put deal{meter}{numeric}, (E_9, 15) put_place stranger{deal}, (E_7, 7)	69
7	extend deal{numeric}{rest}, (E_5, 42) extend deal{numeric}{meter}, (E_9, 16) extend deal, (E_7, 9)	67
8	see_determine_check_view deal{numeric}, (E_5, 33) see_check deal{system}, (E_6, 11) check_ensure_see_determine meter{volume}{deal}, (E_7, 11) see_check deal_trade{numeric}, (E_1, 9)	64
9	send_attach resume{version}{electronic}, (E_3, 30) attach_send_bind resume{detail_point}, (E_1, 16) send_attach resume{interview}, (E_4, 15)	61
10	sell_trade numeric0mw{pricenumeric}{hour0end}{he0numeric}, (E_1, 58)	58
11	allocate volume0management{deal}{contract}{meter_mtr}, (E_5, 46) allocate deal, (E_7, 8)	54
12	purchase_buy gas{plant}, (E_5, 42) purchase_buy gas, (E_6, 7)	49
13	handle_cover deal{numeric}{meter_mtr}, (E_5, 43) handle_cover deal_trade, (E_7, 5)	48
14	set_determine_arrange interview{phone}, (E_3, 27) set_arrange_prepare interview{schedule}, (E_4, 14) put_arrange agenda_schedule, (E_3, 5)	45
15	send_attach spreadsheet, (E_1, 5), (E_5, 7), (E_3, 6) attach spreadsheet, (E_8, 16) attach_send spreadsheet{case}, (E_6, 10)	44
16	forward resume{request}{associate0program}, (E_3, 32) forward resume, (E_4, 11)	43
17	show deal, (E_5, 15) show_designate counterparty{master0net}{agreement}, (E_2, 7) show_indicate credit{trade}, (E_2, 6) show_designate meter, (E_7, 5) show_indicate change, (E_2, 5) show pricenumeric, (E_6, 5)	42
18	invite_receive personnelpronoun{audience_interview}{meeting}, (E_3, 42)	42
19	attach_transmit_send agreement{sheet}{discussion}{cover}, (E_2, 24) attach_tie_send draft{voicemail}{sale0agreement}{notice}, (E_2, 14) send draft{duffie}{plan_program}, (E_3, 7)	41
20	make decision{deal_trade}, (E_1, 20) make decision, (E_3, 9) make deal, (E_5, 9)	38
21	make change, (E_1, 8), (E_5, 21) make change{minor}, (E_3, 8)	37
22	set_arrange_put personname{meeting}{assistant}, (E_3, 33)	33
23	add deal{numeric}, (E_5, 21) add_supply deal{demand0fee}{numeric}{cost_price}, (E_6, 11)	32
24	send_attach_ship transport0contract{deal}{term}, (E_5, 31)	31
25	sell_trade natural0gas{plant}{counterparty}, (E_5, 31)	31
26	schedule locname{zone}{load}, (E_1, 27)	27
27	send_attach info_information{unit}{numeric}, (E_3, 17) send_attach info_information, (E_4, 9)	26
28	make reservation{hotel}{dinner}, (E_3, 16) make reservation{restaurant}, (E_4, 9)	25
29	schedule_reschedule meeting{assistant}, (E_3, 16) schedule_reschedule meeting{eb0numeric}, (E_4, 8)	24
30	wrap_roll deal{numeric}, (E_5, 24)	24

- *create deal{numeric}{ticket}{meter_mtr}* that was discovered from the emails of the employee E5 with a frequency equal to 41;
- *create deal* that was discovered from the emails of the the employee E_9 with a frequency equal to 11;

We highlight (in blue color) the most frequent activity name belonging to each activity group and presenting its final label. Finally, we provide the overall frequency of each activity group (N_{Freq}). To understand the business aspect of the discovered activities, we discuss them according to the following categories:

- Activities of trading power energy: buying/selling electricity power (e.g. ID=10, ID=30, in Table 5.5);
- Activities of trading gas energy: buying/selling gas energy (e.g. ID=12), managing gas deals, e.g. create deal, i.e. ID=3; set/adjust/change/revise/check the values of its attributes (price, counterpart organization, trading dates), i.e. ID $\in \{4, 6, 8\}$; extend deal deadlines, i.e. ID=7 ;
- Activities of organizing interviews: send/forward resume, i.e. ID $\in \{9, 19\}$; conduct/ schedule/ arrange interviews, e.g. ID $\in \{5, 14\}$;
- Standard activities (that could be performed in various business contexts): attend meeting, schedule meeting, send files/information, prepare/send presentation;
- Activities of sending legal documents: send master agreement contracts, send transport contract;
- Activities of publishing papers: attend conference, write papers, send/review comments, send paper drafts;

Table 5.5 additionally shows how similar activities were mostly well grouped. This is confirmed by the adjusted random index (i.e. $ARI = 0.77$) that we calculated for assessing the quality of grouping similar activities of different employees (after manually annotating each discovered activity per employee in terms of the global activity to which it belongs).

5.4.6 Example of results obtained from Orange employee emails

In this section, we show the effectiveness of our solution for activity discovery when it is applied on emails of another organism than Enron. We present the example of results obtained after analysing a dataset of emails of an Orange employee in a research program manager position.

The analysed dataset contains 2897 emails sent by the research manager during two years (from Oct 03, 2018 to Oct 14, 2020). Compared to the previously analysed data, these emails are mainly written in French language and they could contain ones written in English. To handle such situation, we integrate the following extensions in the activity discovery solution:

- At preprocessing phase, we use the library *langdetect*[6] to detect with each language each email sentence is written. Then, according to the detected language, we use the appropriate library for lematizing and splitting the sentence. For French language, we use the *TreeTagger* library[7] for lemmatization;
- For the synonymy dictionary, we integrate the WOLF (Wordnet Libre du Français, Free French Wordnet) dictionary [8];
- For grouping patterns by main action similairty, WOLF does not provide the derivationally syntactic related terms of words (unlike Wordnet). We remind that this is useful for deducing verb root form of nouns to match them to the discovered verb concepts. To handle such situation, we integrate a multi-language BERT model trained on Wikipedia data [80] and which enables matching similar expressions in terms of meaning and context of use. We use such model for matching each pair of noun and verb having rewording relation while satisfying these two conditions: (i) they have a considerable similarity measure (generated by the BERT model) close to 1, and (ii) they have a common sequence of characters that is relatively large comparing to the two words;

After applying our solution for activity discovery on this dataset, we additionally organise the discovered activities per research projects (that are prior defined by the research manager). This is ensured by associating to each project a set of combinations of key words. If an email contains one of these combination, it will be associated to the corresponding project. Afterwards, we ponder email association to research projects and to activities to deduce the highly correlated activities to each project.

In Figure 5.13, we show examples of results obtained for two research projects (i.e. Project 1 & Project 2) where we hide their real names for confidentiality purposes. For each project, we list the set of activities in a green box (whose labels are mostly discovered in French) and we show the translated English version in the grey box next to it.

For the first project, we could differentiate between two types of activities which distinguishes it from the other project:

- Data oriented activities: These activities act on data useful for conducting the research project, i.e. *'see data'*, *'validate data'*, *'store_provide data'*, *'solicit_request data'*, *'collect data'*. This is logical as such project is actually focused on a data mining task;
- Managing access oriented activities: The research manager performs these activities to manage data access by granting his collaborators (e.g. engineers, interns) the access to useful data or studies, i.e. *'mention access'*, *'send access'*, *'request access{study}'*, *'access study{network}'*, *'access data{intern}'*;

For the second project, we could differentiate between these two types of activities that caracterize it:

[6]https://pypi.org/project/langdetect/
[7]https://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/
[8]http://pauillac.inria.fr/ sagot/index.html#wolf

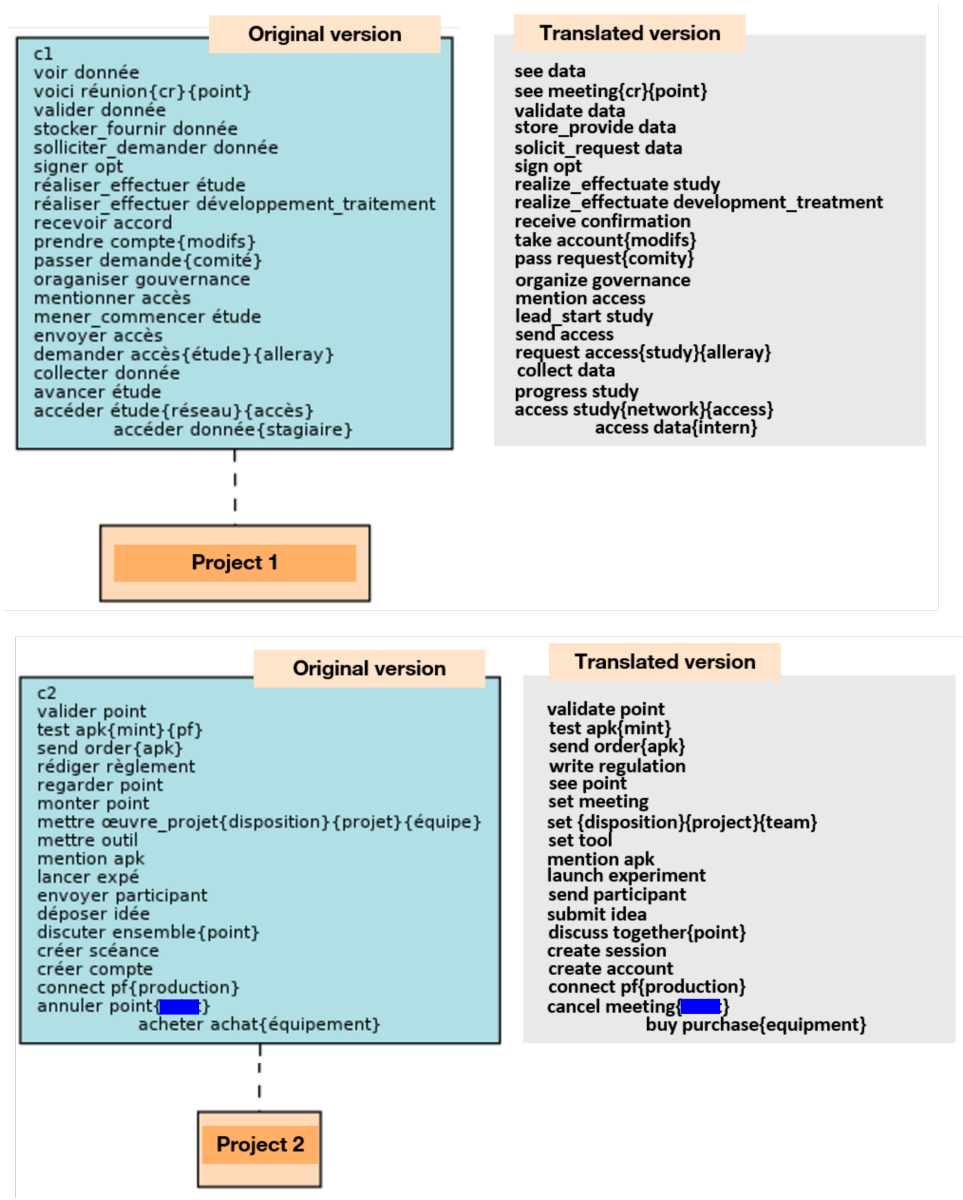


Figure 5.13: Activities per project

- Testing application oriented activities: These activities refer to conducting experiments on a web application, i.e. *'test apk'*, *'send order{apk}'*, *'launch experiment'*. This is understandable as the second research project actually aims at delivering a user application;
- Managing application oriented activities: i.e. *'create session'*, *'create account'*, *'connect pf{production}'*, *'send participant'*;

As for the type of activities that figures in common between the two projects, it is mainly related to managing meeting, e.g. *'set meeting'*, *'cancel meeting'*, *'see meeting{cr}{point}* (the term *'cr'* refers to *'compte rendu'* in French language, which means *'report'* with English). Other activities having organizational nature were additionally discovered such as *'sign opt'*

at the level of the first project (the term *'opt'* refers to *"option'*), *'write regulation'* and *'buy purchase{equipment}* (at the level of the second project).

By consulting the research manager's opinion, he has confirmed that the discovered activities reveal the ones that he frequently performs. He additionally confirmed their relevance degree w.r.t to his job.

5.5 Conclusion

In this chapter, we achieved the first part of the following sub-objective: **Objective 2.1:** Automate the generation of a structured event log from emails. We additionally answered the second question (**Q2**) raised in the thesis problematic (Section 1.2.2), which is : How to discover BP knowledge from emails? w.r.t. activity BP element and according to the following sub-questions:

- Q2-1: How to discover BP elements without disposing a priori knowledge about them ?
- Q2-2: How to select, from emails, relevant information in relation with BP elements ?
- Q2-3: How to discover BP elements considering the different degree of their granularity expression ?
- Q2-4: How to discover BP elements considering their multiplicity in emails ?

To this end, we have proposed an unsupervised approach for discovering frequent activities as well as their business information from emails without disposing prior knowledge about them. We have introduced a pattern discovery algorithm to discover frequent activities discussed through emails. The algorithm is based on several key features:

1. It analyzes per actor sent emails to reduce the variability of expressions. The goal is to capture frequent patterns of words used for expressing activities;
2. It takes into account the words meanings in the generation of the patterns;
3. It considers the business context, defined by the business information that co-occur with patterns in order to regroup them into a single activity.

By characterizing activities in the form of pattern of concepts, our approach allows: (i) the automatic generation of significant labels (names) close to the actual vocabulary of employees, (ii) detecting activities in emails even if they are expressed differently, and (iii) discovering several activities per email if multiple patterns belonging to different activities are present in one email. We have carried also experiments on a public dataset retrieved from Enron and we showed the good performances of our results. Additionally, we have publicly shared our

results (link[3]), which is, in our knowledge, absent in related studies (that's why comparison with them was not feasible when evaluating our proposals).

We agree that some limitations could be derived at three levels:

- At algorithmic level, the results (in terms of activities/patterns) could be affected (in terms of degree of noise) if instance information is expressed in the form of topic of words within emails. Taking the example of activities intended for managing projects, projects (which present here the instance notion) could be introduced by a manager not only through their names, but also through their related topics. Hence, patterns generated at this level could reflect instance information rather than activity names or business context information. This also could affect the quality of grouping patterns into activities types (they would be grouped by instance for example rather than business context information);
- At parameter level, our proposed approach requires several parameters to be set by user. However, we showed through our experiments that these parameters could be logically set to default values (e.g. close to 0.5 for similarity or coexistence thresholds), without the need to vary their values;
- At evaluation level, the email dataset construction for activity discovery mainly depends on the set of employees that are initially selected to use their contact networks for covering other employees. Actually, we have been based during our experiments on one sample of these employees to quantify the approach performances. We agree that more samples should be selected to further investigate their stability. That is why, we started to study them on other employees' emails belonging to a different organism than Enron (i.e. Orange) and written using another language (i.e. French). We showed through the example of results that our approach has the potential of generating relevant activities with these new emails. We will additionally show, in a next chapter (i.e. Chapter 8), other examples of results obtained in different use cases that extend our solution for activity discovery. Nevertheless, we admit that further experimental work must be conducted to quantify the performances of the new obtained results. As for the adopted metrics during the evaluation, we have been mainly based on calculating the relevance or the precision degree of the discovered patterns and activities. The recall was only studied at the level of the discovered activities of two employees. This is due to the considerable human effort required for identifying all the activities that are actually contained in emails. As an alternative, we tried to not be limited to such measure. We assessed the ability of our approach in generating relevant activities with good precision where we considered a larger set of employees (i.e. of size 9). Finally, it is important to note that the manual annotation (necessary for calculating the evaluation metrics) was mainly based on emails without being validated by business experts. This is due to the lack of documentation regarding Enron BP;

In the next chapter, we will focus on detailing the remained algorithmic parts of the second phase of our overall framework aiming for event log generation from email log data.

Chapter 6

Event Log Generation Based on Activities

Contents

6.1 Introduction

In the previous chapter (Chapter 5), we have introduced the first part in the event log generation phase, which aims to discover activities in an unsupervised way from employee emails. In this chapter, we focus on the remained parts in event log generation phase. The goal is to use these activities in order to discover the related events occurring in a set of emails, which are not necessary considered in the learning part (i.e. Part 2.1 in our approach, see Section 5.3). The ultimate goal is to obtain a structured event log compatible with discovering BP fragments w.r.t multiple perspectives.

As formalized earlier (Definition 4.5), each event in our event log corresponds to the occurrence of one activity in an email. It records the following BP elements:

- The occurring activity name, business data and business context information;
- Speech act of the occurring activity name;
- Attributes (i.e. ID, timestamp, conversation ID, sender, recipients) of the email where the activity occurred;
- The set of relevant information values approximating the notion of instance (e.g. see I_{values} in Figure 3.5);
- A tag referring to the thread ID approximating the concept of trace ID in process mining;

In this chapter, we dispose as input: (i) an email log data, (ii) the corresponding preprocessed emails obtained from the previous chapter, and (iii) a learned model of the discovered activities from these emails (obtained from the previous chapter). We apply additional steps distributed on three main parts to obtain a structured event log (see the framed parts with orange color in the framework extract in Figure 6.1). We start by the activity occurrences discovery part (Part 2.2) where we use the characterization of each activity component in terms of patterns (retrieved from the learned model) to discover its occurrences in emails. Then, we extract indications on actors' contributions for each activity occurrence (Part 2.3). This includes: (i) the identification of textual indices that would refer to the performers of activity occurrences, and (ii) the discovery of the speech act associated to each activity occurrence. We introduce for speech act discovery two variants: a rules based approach and a supervised learning approach. In the final part (Part 2.4), based on activity business data occurrences and email attributes, we propose a solution to identify relevant information values approximating the instance concept in process mining. We use these relevant information to construct threads approximating BP fragment traces.

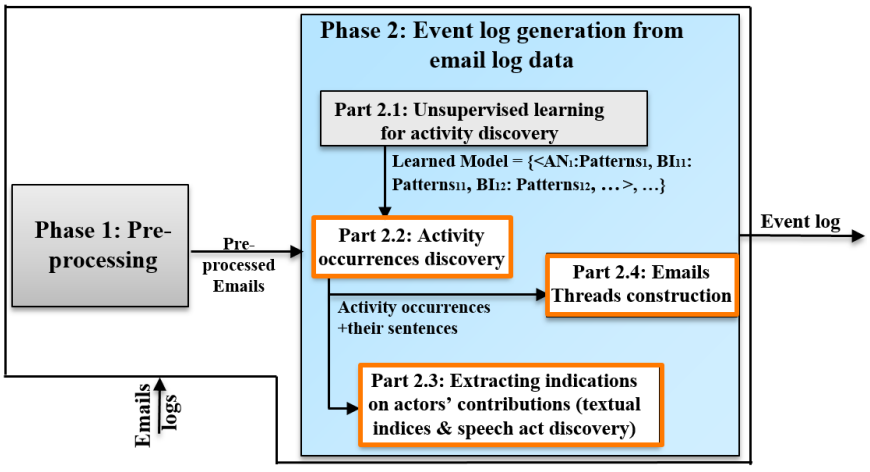


Figure 6.1: Framework extract: Parts to be detailed in Chapter 6

To evaluate the obtained event log, we rely on: (i) extending the real dataset of emails retrieved from the public dataset Enron as described in Table 5.2, and (ii) using the obtained

activities that we describe in Section 5.4.5 and of which we give an overview in Table 5.5. We demonstrate the effectiveness of the obtained results by studying: (i) the F1 score of the speech act discovery step, (ii) The precision of the overall generated event log w.r.t. activity occurrences and speech acts of events, and (iii) the consistency of the constructed threads. We additionally rely during the evaluation of the overall event log generation phase on an experimental tool that we have introduced to: (i) visualize and manage the obtained results in terms of activities, speech acts and the corresponding occurrences, (ii) facilitate their interpretations and their exploration given a considerable number of activities/emails, and (iii) help in the annotation of the discovered BP elements, which is required when carrying out the experimental work.

Some parts in this chapter were published in the International Conference on Process Mining (ICPM) [31].

In the following, we outline in Section 6.2 our solution for activity occurrences discovery in emails. In Section 6.3, we explain our approach for discovering speech acts of activity occurrences. In Section 6.4, we present our solution for email threads construction. Finally and before concluding, we evaluate our proposals in Section 6.5 where we additonally present our experimental tool.

6.2 Activity occurrences discovery

This part discovers the occurrences of activities in emails using their related patterns of concepts. Such patterns are provided from the learned model obtained from the activity discovery part (see Section 5.3). The activity occurrences part aims to: (i) find the exact positions of appearance of activities in emails defining their occurrences, and (ii) extract the sentences where they occurred, which will be useful in the next parts to extract further event attributes. We illustrate in Figure 6.2 the main input/output of such part.

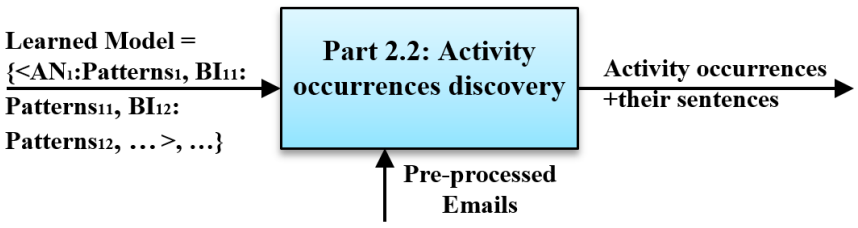


Figure 6.2: Input & Output of activity occurrences discovery

We recall that activities have two types of components: (i) activity names (whose are denoted by $\mathcal{AN}$), and (ii) business information (whose are denoted by $\mathcal{BI}$)) divided into business data (whose are denoted by $\mathcal{BD}$) and business context information (whose are denoted by $\mathcal{BC}$). Definition 6.1, Definition 6.2 and Definition 6.3 formalizes, respectively, the relation between: (i) activities and patterns, (ii) the occurrence of an activity component and an email, (iii) the occurrence of an activity and its component. In what follows, let denote $\mathcal{R}_f$

the set of raw words as they appear in emails before lemmatization.

Definition 6.1. ***(Mapping-To-Patterns)*** *A Mapping-To-Patterns function is defined as* $\mathcal{M}_{Pat} : \mathcal{AN} \cup \mathcal{BD} \cup \mathcal{BC} \rightarrow \mathcal{PC}^*$ *where* $\mathcal{M}_{Pat}$ *returns the list of patterns of concepts associated to an activity component type (*$\in \mathcal{AN} \cup \mathcal{BD} \cup \mathcal{BC}$*).*

Definition 6.2. ***(Activity Component Occurrence)*** *Let consider the following notations:*

- $em \in \mathcal{E}$ *be a preprocessed email;*
- $AC \in \mathcal{AN} \cup \mathcal{BD} \cup \mathcal{BC}$ *be an activity component;*
- $PC \in \mathcal{PC}$ *be a pattern such that* $PC \in \mathcal{M}_{pat}(AC)$*;*
- $f_{Pos} : \mathcal{PC} \times \mathcal{E} \rightarrow (\mathbb{N}^*)^*$ *be the function that returns the set of the overall combinations of appearance positions of the concepts of a given pattern* $\in \mathcal{PC}$ *in a preprocessed email* $\in \mathcal{E}$ *(Each one of these combinations is sorted in ascending order).*
- $f_{Rf} : \mathcal{E} \times \mathbb{N} \rightarrow \mathcal{R}_f$ *be the function that returns for a lemmatized word, appearing at a given position* $ps \in \mathbb{N}$ *in a preprocessed email* $em \in \mathcal{E}$*, the corresponding raw form. This means that:* $f_{Rf}(em, ps) = em[ps].w_{Rf}$

Let $AC_{occ} = (AC, L_{Rf}, Pos_{AC})$ *be the occurrence of the activity component* AC *in the email* em *w.r.t to the list of raw words* $L_{Rf} \subset \mathcal{R}_f$ *appearing in the positions* $Pos_{AC} \subset \mathbb{N}$*.* AC_{occ} *verifies the following conditions:*

(i) $\exists\ PC \in \mathcal{PC}$ *such that* $PC \in \mathcal{M}_{pat}(AC)$*,* $f_{Pos}(PC, em) \neq \emptyset$ *and* $Pos_{AC} \in f_{Pos}(PC, em)$*;*

(ii) $\forall\ C = (L_w, t) \in PC,\ \exists\ w_{lem} \in L_w \mid w_{lem} \in LEM_P(em)$*;*

(iii) $\forall\ i \in [1, |Pos_{PC}| - 1],\ |Pos_{PC}[i+1] - Pos_{PC}[i]| \leq Th_d \mid Th_d \in \mathbb{N}$*; AND*

(iv) $L_{Rf} = [f_{Rf}(em[ps]),\ ps \in Pos_{AC}]$

An activity component occurs in an email $em \in \mathcal{E}$ if: (i) one of its patterns is detected, and (ii) words referring to the occurrence of its concepts are low dispersed. Low dispersity of words (ensured by a defined threshold Th_d) means that they appear close to each other in an email em. This guarantees that if a pattern is detected in the email em, it would express the same activity component objective.

Definition 6.3. ***(Activity Occurrence)*** *Let* $Act = (AN,\ BD,\ BC) \in \mathcal{A}$ *be an activity as defined in Definition 4.2. Let* $f_{occ} : (\mathcal{AN} \cup \mathcal{BD} \cup \mathcal{BC}) \times \mathcal{E} \rightarrow \mathcal{O}^*$ *be a function that returns the occurrences of an activity component in an email* $em \in \mathcal{E}$*. The activity* Act *has an occurrence* $Act_o = (AN_{occ}, BD_{occ}, BC_{occ}) \in \mathcal{O} \times \mathcal{O}^* \times \mathcal{O}^*$ *in the email* em *if and only if:*

(i) $f_{occ}(AN, em) \neq \emptyset$

(ii) $AN_{occ} \in f_{occ}(AN, em)$

(iii) $BD_{occ} = \{f_{occ}(B, em), bd \in BD\}$; *AND*

(iv) $BC_{occ} = \{f_{occ}(bc, em), bc \in BC\}$

An occurrence of one activity is considered discovered in an email if mainly the occurrence of a pattern representing the corresponding activity name is discovered. In fact, business context information or business data values would not always be present in the same email.

To discover the occurrences of each activity component $AC \in AN \cup BD$ having a set of patterns $L_{PC} = \mathcal{M}_{pat}(AC)$, we iteratively search all the occurrences of each pattern ($\in L_{PC}$) in each email while verifying the low dispersity criterion. First, we replace words forming patterns' concepts in a preprocessed email by tags reflecting them. The part of speech tag of these words in the email must match those of the patterns' concepts. After that, we reduce the preprocessed email to the list of concepts' tags while keeping their real positions of appearance (without removing those of redundant words). Therefore, we sort the elements of this list in an ascending order according to these positions. By scrolling through this list, we add each element to the same pattern occurrence if: (i) its tag was not previously added, (ii) the position distance that separates it from the last added element does not exceed a defined threshold (Th_d), and (iii) it is assigned to the same sentence of the tag previously added. Once having an element where (ii) or (iii) is not verified, this latter marks the end of one potential pattern occurrence and the beginning of another.

Let consider the example composed of these elements:

- The preprocessed format $List_{TUP}$ of the email $email_2$ (of Figure 1.1) where $List_{TUP} = [$ ($'Flow'$, 0, $'flow'$, $'n'$, 0), ($'nom'$, 1, $'nom'$, $'n'$, 0), ($'Rolled'$, 2, $'roll'$, $'v'$, 1), ($'deal'$, 3, $'n'$, 1), ($'454057'$, 4, $'numeric'$, $'n'$, 1), ($'cover'$, 5, $'cover'$, $'v'$, 1), ($'flow'$, 6, $'flow'$, $'n'$, 1), ($'mtr'$, 7, $'mtr'$, $'n'$, 1), ($'5192'$, 8, $'numeric'$, $'n'$, 1)].

- The activity $Act_1 = (AN_1, \{bd_{11}, bd_{12}\}, \{\})$ such that :
  - $PC_{AN_1} = \{(\{'roll'\},' v'), (\{'deal'\},' n')\} \in \mathcal{M}_{pat}(AN_1)$;
  - $PC_{bd_{11}} = \{(\{'deal'\},' n'), (\{'numeric'\},' n')\} \in \mathcal{M}_{pat}(bd_{11})$;
  - $PC_{bd_{12}} = \{(\{'mtr'\},' n'), (\{'numeric'\},' n')\} \in \mathcal{M}_{pat}(bd_{12})$;

In the case of of the pattern PC_{AN_1} ($PC_{bd_{11}}$, $PC_{bd_{12}}$), the words {*'roll'*, *'deal'*} ({*'deal'*, *'numeric'*}, {*'mtr'*, *'numeric'*}) appear close to each other and in the same sentence in the preprocessed format of $email_2$. This induces the detection of the activity component AN_1 (bd_{11}, bd_{12}) occurrence having the following expressions: AN_{1occ} = (AN_1, [$'rolled'$, $'deal'$], $[2, 3]$) (bd_{11occ} = (bd_{11}, [$'deal'$,$'$ 45057$'$], $[3, 4]$), bd_{12occ} = (bd_{12}, [$'mtr'$, $'5192'$], $[7, 8]$)).

6.3 Extraction of indications on actors' contributions

This part extracts, from the activity occurrences' sentences, a set of indications that would refer to the actual contributions of emails' interlocutors towards the occurred activities. These indications are of two types: (i) textual actors' indices (e.g. personnel pronouns) that would refer to the performer(s) of the occurred activity(ies), and (ii) speech acts of activity occurrences that are necessary for inferring their contributions and for deducing some event types. To this end, it performs as illustrated in Figure 6.3 three main steps as follows. In the first step, it extracts a set of local features, related to activity occurrences, from the emails where they occurred. Then, it uses these features to extract actors' indices in the second step and to discover activity occurrences' speech acts in the third step. In what follows, we explain each one of these steps.

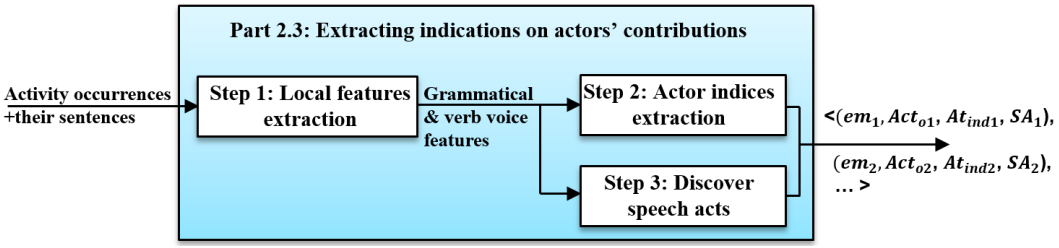


Figure 6.3: Extraction of indications on actors' contributions: Main steps

6.3.1 Step 1: Local features extraction

This step uses the entire sentence where an activity name occurred to extract local features related to its verb. These features are of two types; (i) grammatical, and (ii) verb voice, i.e. passive or active.

Grammatical features are obtained after **parsing grammatical dependencies** [41] from the sentence (without removing standard stop words as they would be useful). This returns a list of tuples where each one corresponds to one word. It has three additional elements comparing to the preprocessed email structure defined in Definition 5.1:

- ***Specified part of speech tag of the word***: Comparing with what was previously adopted, the notion of part of speech tag of the word in this section do not only indicates if it is a verb or not. It additionally indicates its grammatical category such as word's tense (e.g. past participial, verb ING) or number (plural/singular);
- ***Word's dependency tag***: This tag defines the grammatical relation between the word and the verbal element to which it depends, e.g. *'subject'* (*'agent'*) is a dependency tag between the action's performer and the verb in active (passive) voice;
- ***Word's Subtrees***: A subtree of a word w refers to the subset of words having direct grammatical dependencies with w, e.g. {*'that'*, *'was'*, *'purchased'*, *'for'*, *'bal-day'*} forms

the subtree of *'purchased'* in the following sentence: *'This deal would need to be cut and replaced with the power that was purchased for bal-day, etc.'*;

For identifying, for each activity occurrence, **the voice of the verb** being used to express it, this step uses verb's subtree to check if it verifies the specific grammatical structure of a passive voice sentence. Otherwise, the verb voice is considered as active. In this work, this specific structure is considered as the composition of: (i) an auxiliary dependency tag referring to an auxiliary verb (*'be'* or sometimes *'get'*), and (ii) a specified part of speech tag referring to the past participle of the main verb denoting the action. We could denote this structure as follows: *auxiliary verb+past participle*.

6.3.2 Step 2: Extracting actors' indices

Actors indices would be useful to define the actual contributions of some emails' interlocutors. In fact, if the sender uses the first (or the second) personnel pronoun to express the doer of the activity action, it is an indication that he (or some recipients) is (are) the actual executor(s). This step studies two cases:

(i) If the verb of the activity is in passive voice: The doer of the action may be specified in what follows the activity verb using a prepositional phrase with the preposition *'by'* e.g. *'it was purchased yesterday by Mark'*. This steps localizes then such prepositional phrase in case it was indicated;

(ii) If the verb of the activity is in active voice: The doer of the action may be specified by the subject of the sentence subtree of the activity verb. The position of such subject is then determined in order to find its correspondent textual content;

6.3.3 Step 3: Discover speech acts

This step focuses on discovering the speech acts related to the occurrences of activities in emails. More precisely, it aims to identify, for each activity occurrence, the speech act of the verb used for expressing the corresponding activity name. We recall that we consider four speech acts: request act, request information act, intention act and information act.

Towards this goal, we propose two methods; (i) ***rules based method***: It is based on injecting a human expertise to define a decision tree summarizing some rules derived form natural language, and (ii) ***supervised method***: It applies any existing supervised learning algorithm. It does not require a human expertise to define discovery rules. However, it leverages a labeled training dataset.

These two methods are based on analysing four kinds of properties of each verb characterizing an activity occurrence;

(i) *Grammatical properties:* They concern the verb's voice, form (in basic form or not), specified part of speech tag and dependency tag;

(ii) *Subtree properties:* They concern words (basic form & grammatical properties) preceding the verb in its subtree;

(iii) *Neighbourhood properties:* They concern words preceding the verb and appearing close to it in the same sentence. The closeness of a word is defined by a distance threshold separating its position of appearance from that of the verb;

(iv) *Sentence properties:* They concern some specific words (e.g. 'I', *'we'*, *'you'*, *'please'*, etc) or punctuation ('?') appearing in the overall sentence of the verb;

In the case of **the supervised method**, for each occurred activity, learning features are deduced from the specified part of speech tags and the dependency tags of: (1) the verb used to express it, and (2) the words belonging to the sentence where it occurs. The learning features additionally include some specific words indicating tense (e.g. *'yesterday'*), personnel pronoun (e.g. 'I', *'we'*, *'you'*), request words (e.g. *'please'*) and sentence punctuation. As they must have numerical representations during the learning phase, this method calculates the distance that separates them from the activity verb (in terms of absolute value of difference between their positions of appearance and that of the verb). In the case of non existence of one feature, a large distance value is considered.

In the case of **rules based method**, it aims to identify an activity occurrence speech act on the basis of the tense of its verb (that could be deduced from its specified part of speech tag). In fact, if emails' senders use an activity verb in past, they would inform about its execution (e.g. *'I created a ticket'*). If they use it in future, they would express an intention of execution. As for verb in present or in basic form, additional properties (e.g. neighbourhood) must be analysed to identify the correspondent speech act. This method defines then a set of rules to ensure tense-speech act mapping and summarizes it in the form of a decision tree (Figure 6.4). Our tree is composed of ten levels (discontinuous rectangles, e.g. L1, L2). Each level could contain; (i) a diamond shape referring to a verb property to be verified, e.g. D1: 'Verb voice is active' (ii) Rounded shape indicating tree's end terminators while precising the discovered speech act (SA), e.g. SA=Request, and (iii) Rectangle shape referring to an action to be done after a decision, e.g. A2: Return to D1. Levels of our tree are of three categories:
- *Sentence form decision levels (L1&L2):* These levels concern these two types of features:

(i) Sentence punctuation (L1): if it is a question, the speech act will be a request information act unless the verb is preceded by a specific modal verb (see MD1 in the abbreviations of Figure 6.4, e.g. can, could, would) to be considered as a request act;

(ii) Verb voice (L2): if it is in passive voice, the speech act is assigned to an information act unless the auxiliary verb is not expressed in its basic form (i.e 'be'). Otherwise, the verb is replaced by its auxiliary verb;

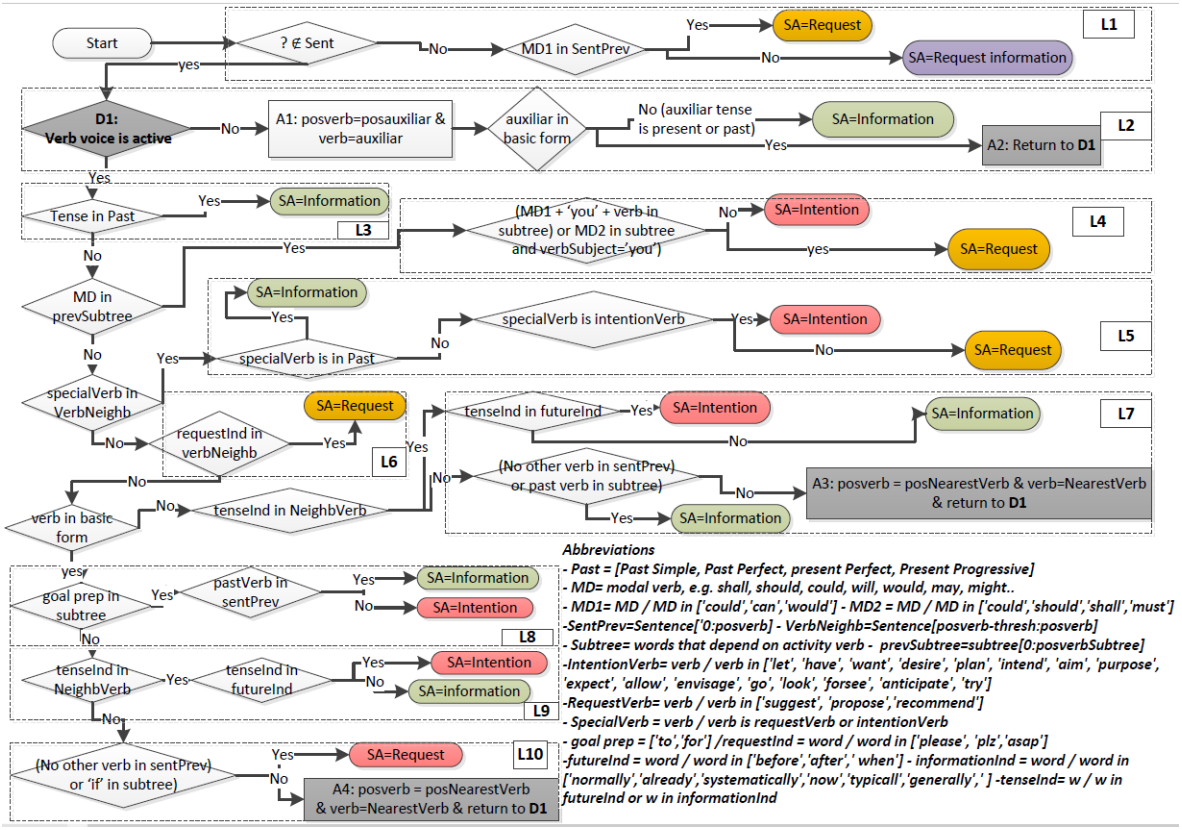


Figure 6.4: Speech Acts Prediction Tree

- *Own and neighborhood decision levels (L3 → L6):* Starting from these levels, the verb is considered in an active voice and some own and neighbourhood properties are checked. The closer they are to the verb, the higher the priority of the corresponding rules/decisions. As verb's part of speech tag is an own property, it has a higher priority. This tag could directly indicate if the verb is in 'Past' tense or not to recommend if the speech act is an information act (L3). Then, the priority passes to some specific words appearing in the verb neighborhood:

(i) Modal verbs in L4 (see MD in the abbreviations of Figure 6.4), e.g. will, would, shall: They recommend that the speech act is an intention act unless 'you' is present in verb subtree as a subject or switched with some specific modal verbs (see MD2 in the abbreviations of Figure 6.4, e.g. can, would);

(ii) Some special verbs (in L5) of intention (e.g. want, plan) or of request (e.g. suggest): They recommend that the speech act is an intention or request act unless they are expressed in 'Past' (the speech act will be an information act);

(iii) Some special words indicating a request speech act (*'requestInd'*, e.g. 'please') in L6;

- *Transitive decision levels (L7 → 10):* Here, verb is considered either: (i) a third-person verb, or (ii) in basic form ($L8 \rightarrow L10$). Verb neighborhood would be not sufficient to recommend the corresponding tense or speech act. Some properties are then checked to transfer the speech act identification to the closer verb preceding the activity verb. This operation is repeated

unless no verb precedes the activity verb or the speech act is predicted. If at the end, the speech act is not yet predicted, it will be assigned to information act (L7) or request act (L10) in case of a third-person verb and a verb in basic form respectively.

6.4 Email threads construction

This part organizes emails into threads, so that communication traces related to the same instance could be collected in the same structure. We recall that a thread refers to a set of conversations grouped by the mean of relevant information values approximating the notion of instance. As illustrated in Figure 6.5, this part is divided into two steps. In the first one, we have to group emails into conversations (Section 6.4.1). In the second one, we have to determine the relevant information that could define instance notion to be selected and used for conversation assembly into threads.

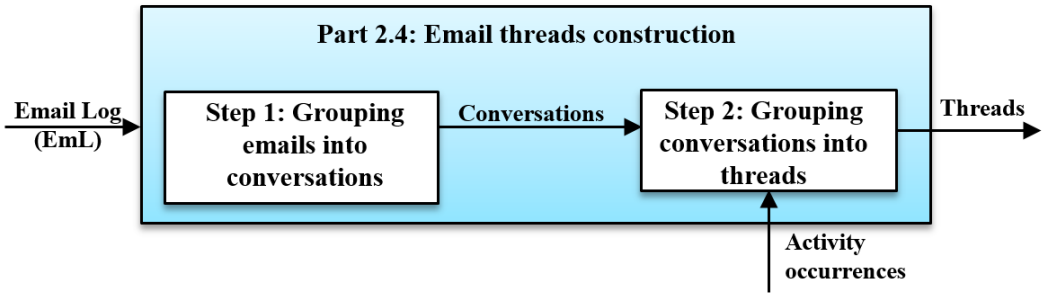


Figure 6.5: Email threads construction: Main steps

6.4.1 Step 1: Grouping emails into conversations

The goal of this step is to organize raw emails ($\in EmL$) into conversations, which is useful for constructing email threads. A conversation in our work is formally defined as follows (Definition 6.4):

Definition 6.4. *(Email Conversation) An email conversation conv is a set of l emails* $\{email_1, .., email_l\} \subset EmL$ *such that; if* $l > 1$*, then,* $\forall\ i \in [1, l]$*,* $\exists j \in [1, l] \setminus \{i\}$ *where* $email_i$ *and* $email_j$ *have a reply or forward relation.*

A conversation regroups then a set of emails having reply/forward relations. In this step, each email is represented by its textual content composed of: (1) an email main body, and (2) its conversational history. Actually, in the context of conversations, each email can be viewed as: (1) a reply or a forward to at most one previous email, or/and (2) replied or forwarded in zero or multiple next emails. Figure 6.6 further details such relations between an original email (*Em*) and its previous (*EmPrev*) and next emails {*EmNext1*, *EmNext2*,..}. It shows also textual content relations between them: for one email (*Em*), its conversational history is

the same as the body textual content of its previous email. Equally, conversational histories of its next emails are the same as its body textual content.

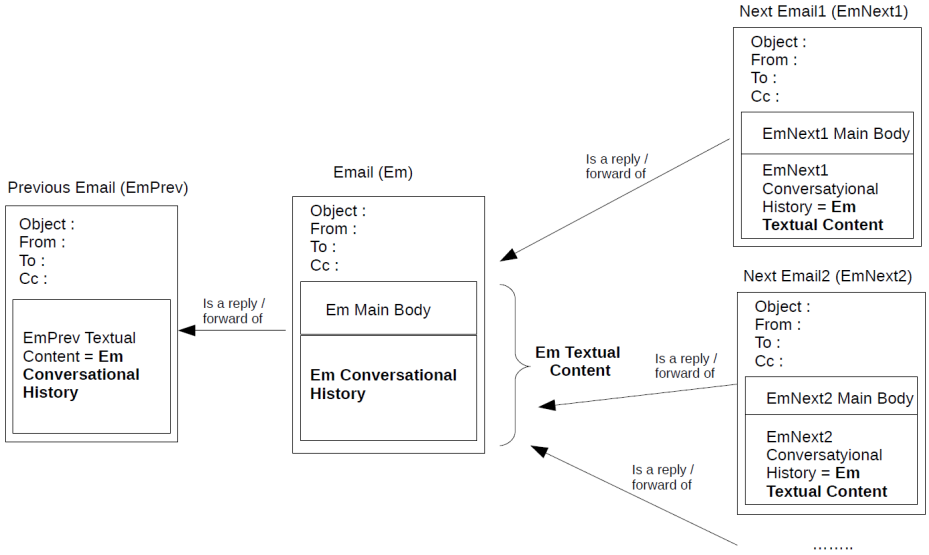


Figure 6.6: Textual content relations between emails of the same conversation

Using such characterization in terms of textual content relations between emails of the same conversation, our solution for constructing conversations is composed of two main sub-steps:

Sub-Step 2.1: Search previous/next emails related to each email: For each email, this sub-step uses its overall textual content to check if it matches the conversational history of other emails. In this way, next emails sent as a reaction (i.e. reply, forward) to the actual email will be identified. To find previous related emails, this step uses the conversational history of each email to check if it matches the textual content of other emails.

Sub-Step 2.2: Iteratively construct conversations: This step goes through all emails list. For each email, if it was not previously affected to existing conversations, it searches its previous and next emails. Then, for each previous (next) email, it recursively searches its own previous (next) email(s) until there is no more previous (next) reaction(s).

6.4.2 Step 2: Grouping conversations into threads

This step regroups conversations having relevant information values in common. To formally define a relevant information value approximating instance notion (see Definition 6.5), let $\mathcal{RC}$ denote the set of recipients, let To denote the information of receiver type with *'to'* status (see Definition 4.1) and let $\mathcal{O}_{\mathcal{BD}}$ denote the set of occurrences values of business data. Each occurrence $\in \mathcal{O}_{\mathcal{BD}}$ of a business data $bd \in BD$ combines bd and its value of occurrence (e.g. 'meter numeric_meter 1601' is an occurrence value of the $bd =$*'meter numeric'* in the email $email_1$ of Figure 1.1).

Definition 6.5. ***(Relevant information value)*** *Let* $f_{inst} : (\mathcal{BD} \cup \{To\}) \times \mathcal{E} \rightarrow \mathcal{O}_{\mathcal{BD}}{}^* \cup \mathcal{RC}$ *be a function that returns the instantiation values of an information type* $\in \mathcal{BD} \cup \{To\}$ *in emails.* RI *is a relevant information* $\Leftrightarrow \exists I \in \mathcal{BD} \cup \{To\}$ *while;* $RI \in f_{inst}(I)$*, AND* $\frac{card(Unique(f_{inst}(RI)))}{card(f_{inst}(RI))} \geq th_{ri}$ *where* $th_{ri} \in [0.5, 1]$ *is a threshold defining the degree of value uniqueness when instantiating* RI*.*

Relevant information value could be of two categories: (1) business data values, and (2) email addresses of activity email receivers. They refer in our work to values of information types ($\in \mathcal{BD} \cup \{To\}$) having largely distinct instantiation values. This is ensured by comparing for each information type: (i) the ratio between the number of its unique instantiation values and the number of its overall instantiation values, with (ii) the threshold th_{ri}, which must be logically greater than 0.5 to guarantee that the overall instantiation values tend to be unique.

Let $\mathcal{RI}$, $\mathcal{CV}$ and $\mathcal{TH}$ denote the sets of relevant information values, email conversations and threads of emails respectively. A thread is formally defined as follows:

Definition 6.6. ***(Thread)*** *Let* $f_{convinst} : (\mathcal{BD} \cup \{To\}) \times \mathcal{CV} \rightarrow \mathcal{O}_{\mathcal{BD}}{}^* \cup \mathcal{RC}$ *be a function that returns the instantiation values of an information type* $\in \mathcal{BD} \cup \mathcal{RC}$ *in a conversation* ($\in \mathcal{CV}$)*. Let* $f_{contact} : \mathcal{E} \rightarrow \mathcal{AD}$ *be a function that returns the set of interlocutors addresses (sender & receivers with different status)* $\in \mathcal{AD}$ *of an email* $\in \mathcal{E}$*.*
A thread of emails is a set of p conversations $\{cv_1, .., cv_p\} \subset \mathcal{CV}$ *while;*

(i) $\exists\ RI \in \mathcal{O}^*_{\mathcal{BD}} \cup \mathcal{RC}\ /\ \{\bigcap_{j \in [1,p]} f_{convinst}(RI, cv_j) \neq \emptyset\}$*;*

(ii) $\{\bigcap_{j \in [1,p]} f_{contact}(cv_j)\} \neq \emptyset$*;*

A thread is a set of conversation that must have in common at least one relevant information value (i.e. (i) in Definition 6.6) and one interlocutor (i.e. (ii) in Definition 6.6). The intersection in terms of interlocutors between conversations would guarantee that threads do not diverge to different actors groups handling the same relevant information value for different purposes.

To obtain these threads, we proceed as follows. At each activity occurrence, business data values and email addresses mentioned in the related email are filtered and only are kept those of relevant information values. Then, all conversations (Definition 6.4) mentioning them (at the level of their emails textual contents or their interlocutor lists) will be collected and associated to the same thread of the activity occurrence's email. This operation is performed while ensuring that conversations have at least one interlocutor in common.

To minimize the possibility that activities of one thread would be related to different instances, only one relevant information value is considered to construct each thread. This means that if one email contains a number n of relevant information values, n threads would be constructed and only if they contain the same emails, they will be merged.

At the end of this step, one thread will be organized in the form of a set of conversations sorted in ascending order according to their starting date and redundant threads will be removed.

6.5 Evaluation

This section evaluates the results of the remained parts belonging to the event log generation phase related to: (i) activity speech acts discovery (Section 6.5.2) and (ii) the overall generated event log including activity and speech act occurrences and the constructed threads (Section 6.5.3).

During this section, we first outline our methodology for dataset construction. Afterwards, we present first our experiment results (Section 6.5.2 and Section 6.5.3). Then, we talk in Section 6.5.4 about the experimental tool that we have used to visualize the results of some parts of the event log generation phase in order to facilitate the annotation and the interpretation tasks.

6.5.1 Email dataset construction for event log generation

For evaluating the overall event log generation phase, we extend the obtained dataset of activity discovery (described in 5.2) to contain emails: (i) sent by the same set of nine employees (used for activity discovery as described in 5.2), (ii) sent by other employees that frequently receive activities' oriented emails, and (iii) appearing in the same threads of activities' related emails. Therefore, we apply the remained parts in the event log generation phase on: (i) the overall obtained emails, and (ii) the obtained activities that we describe in Section 5.4.5 and of which we give an overview in Table 5.5.

Table 6.1 summarizes the features of our evaluation dataset for event log generation, extending Table 5.2. For each employee, we report additional attributes having the following

Table 6.1: Evaluation Dataset For Event Log Generation

Ep	***OR***	***BR***	$\mathbf{N_{em}}$	$\mathbf{N_{emRe}}$	$\mathbf{N_{Re}}$	$\mathbf{N_{emTh}}$	$\mathbf{N_{reTh}}$
E1	Managing Director	Trading	343	421	18		
E2	Senior Counsel	Legal	102				
E3	Managing Director	Risk	2283	504	8		
E4	Assistant	Management	357				
E5	Manager	Logistics	738			1180	95
E6	Specialist	Settlements	108				
E7	Specialist	Logistics	100	682	31		
E8	Employee	Employee	158				
E9	Specialist	Logistics	80				

significations: N_{Re} refers to the number of the selected receivers for each set of employees. N_{emRe} indicates the number of emails that were sent by them and considered in our analysis. N_{th} corresponds to the number of emails (of non empty main body) considered due to their appearance in the same threads with activities related emails. Finally, N_{reTh} reports the total number of additional employees that were present in these threads as email senders.

6.5.2 Speech acts discovery

We evaluate in this part our speech act discovery methods. To this end, we first discovered the occurrences of the activities obtained at the level of the unsupervised learning part in our approach. Then, we selected a sample of 937 activities' occurrences (mainly from emails of E1, E3 and E4) to construct an evaluation dataset for speech act discovery. Afterwards, we annotated each activity occurrence according to its speech act (i.e. information, request, request information or intention). Finally, we applied our *rules based method* on the overall annotated data. As for *the supervised method*, we used the following supervised learning algorithms to test its performances; SVM (Support Vector Machine), RF (Random Forest) and DT (Decision Tree). Then, we applied the stratified[1] k-fold cross-validation method (where k=4) to evaluate their performances.

To compare the discovered speech acts with the annotated ones, we use **F1-score** metric. We summarize the obtained results in Table 6.2. For each speech act, we indicate the total of annotated data (N). We give the obtained F1-scores of the rules based method ($f1_r$), SVM ($f1_{SVM}$), RF($f1_{RF}$) and DT ($f1_{DT}$) after applying each approach on the train/test dataset. Each F1-score measure is indicated in the form of $AVG \pm std$ where: (i) AVG refers to the average value of the F1-scores obtained from the different k-fold partitions, (ii) *std* refers to their standard deviation. As noticed, the two variants of our speech act classification methods tend to have good scores ($\in [0.8, 0.91]$). As rules based method obtained the higher scores, we adopt its results for the remained steps.

Table 6.2: F1-Scores

SA	***N***	***Category***	$f1_r$	$f1_{SVM}$	$f1_{RF}$	$f1_{DT}$
intention	416	train	0.92 ± 0.007	0.85 ± 0.008	0.91 ± 0.008	0.85 ± 0.014
		test	0.92 ± 0.03	0.81 ± 0.054	0.83 ± 0.04	0.81 ± 0.047
information	333	train	0.91 ± 0.005	0.85 ± 0.011	0.9 ± 0.009	0.84 ± 0.016
		test	0.91 ± 0.02	0.8 ± 0.051	0.81 ± 0.055	0.8 ± 0.045
request	132	train	0.9 ± 0.01	0.84 ± 0.013	0.93 ± 0.003	0.85 ± 0.022
		test	0.9 ± 0.05	0.8 ± 0.066	0.84 ± 0.06	0.81 ± 0.046
request information	56	train	0.89 ± 0.02	0.88 ± 0.085	0.96 ± 0.015	0.85 ± 0.016
		test	0.89 ± 0.1	0.81 ± 0.05	0.82 ± 0.138	0.81 ± 0.074
Total	**937**	train	$\mathbf{0.91 \pm 0.006}$	$\mathbf{0.85 \pm 0.009}$	$\mathbf{0.91 \pm 0.006}$	$\mathbf{0.85 \pm 0.014}$
		test	$\mathbf{0.91 \pm 0.028}$	$\mathbf{0.81 \pm 0.05}$	$\mathbf{0.82 \pm 0.04}$	$\mathbf{0.8 \pm 0.04}$

6.5.3 Generated event log evaluation

This part describes the obtained event log and evaluates its quality after using the previously discovered activities as input to our second phase (i.e. event log generation). To assess the event log quality, we were mainly focused on calculating the: (i) activity discovery and speech act discovery precision, and (ii) the consistency of the constructed threads approximating the trace concept in process mining. In what follows, we start by outlining some related

[1]the stratified k-fold cross-validation preserves the imbalanced class distribution in each fold

statistics/features (e.g. size, distribution, activity occurrences and speech acts' precision) that describes it and that resumes its overall quality (Section 6.5.3.1). Then, we detail the precision variation of activity occurrences and speech acts discovery w.r.t. activities and employees (Section 6.5.3.2). We provide the overall event log that we obtained (in json file format) in in this link[2] (see the *Event Log Generation* section in this link).

6.5.3.1 Overall features of the generated event log

We outline the BP element features of the generated event log in Table 6.3. For each BP element (i.e. activities, activity occurrences, speech acts, threads and relevant information values), we report three features:

Table 6.3: Event log features

BP element	***Number***	***Metric value***	***Distribution***
Activities	102	Relevance = 0.93	–
Activity occurrences	3102	Precision = 0.85	Per activity sublog size: see Figure 6.7
Speech acts	–	Precision=0.88	Occurrence number and precision per speech act: (Information, 1296, 0.93) (Intention, 849, 0.84) (Request, 350, 0.85) (Request information, 157, 0.77)
Threads	1287	Consistency = 0.85	Thread number per size: see Figure 6.8
Relevant information values	194		(business data values, 86), (email address, 103)

(i) Number: e.g. number of activities/ activity occurrences;

(ii) Metric value: we report metric measures that reflect the quality (e.g. precision) of the discovered BP elements in the overall event logs. For activities, we calculate the relevance ratio which corresponds to the number of significant activities divided per the total number of the discovered ones. For activity occurrences and their speech acts, we calculate their precision ratio (i.e. number of correct discovered activity occurrences/speech acts divided by the number of their overall occurrences). As for threads and relevant information values, we calculate a consistency measure that reflects to what extent each discovered relevant information value has contributed to reassembling emails belonging to only one instance and sharing the same business context in the same thread. To this end, we first extracted all the relevant information values that induced the regrouping of at least two conversations in the same thread. Then, we consulted the related threads and we manually identified for each one, the number of business contexts/instances whose emails are in relation. Finally, for a set of annotated numbers An, we used the following expression to quantify the consistency of the discovered threads:

$$Consistency(An) = \frac{\sum_{n \in An} \frac{1}{n}}{|An|} \tag{6.1}$$

[2]http://www-inf.it-sudparis.eu/SIMBAD/tools/processDiscoveryFromEmails/

This expression considers the average of the inverse of the annotated numbers per threads. In other words, for one thread, the fewer instances or business contexts related to its emails, the more it contributes to increasing the consistency coefficient. It is important to note that we have not considered threads containing one email or whose emails are related to only one conversation. In fact, relevant information values do not intervene in regrouping their emails, which is not adequate with what we aim to evaluate from our consistency measure.

(iii) Distribution: We describe (or refer to a Figure that shows) the distribution of a BP element feature (e.g. number, precision) by a chosen criterion. For activity occurrences, we show their distribution per activity sublog size in Figure 6.7. For speech acts, we report a tuple for each type (i.e. information, intention, request and request information) of the following format: (type, number of occurrences, precision). For threads, we show the distribution of their number per size (i.e. number of emails). As noticed, a good part of our threads contain a single email (i.e. 789 threads). The rest contains more than two emails (i.e. 498 threads) regrouping a total number of emails equal to 2544.

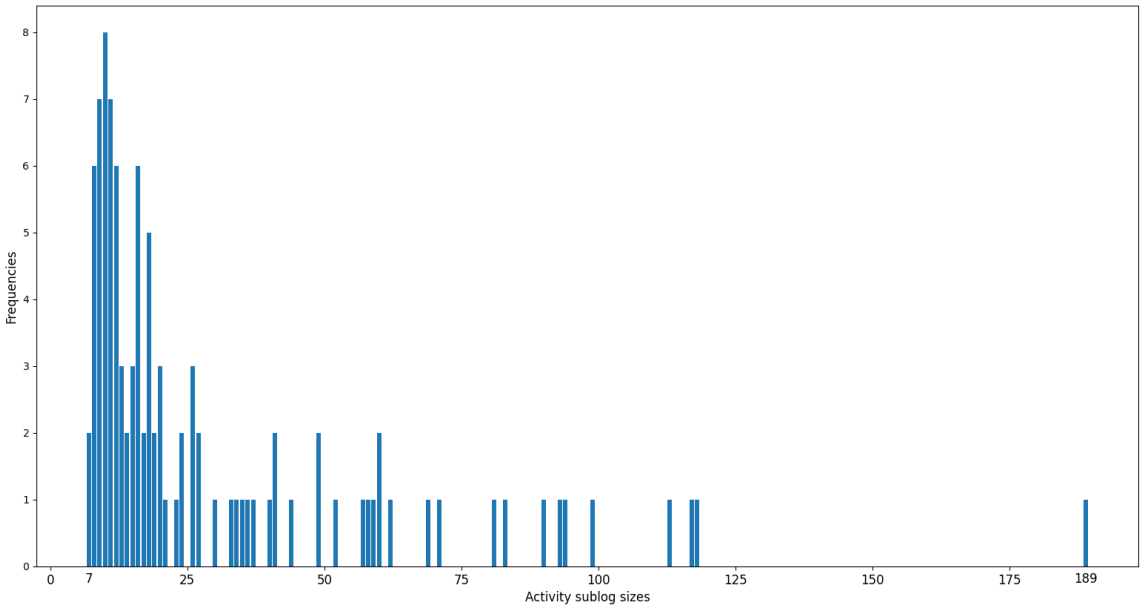


Figure 6.7: Distribution of activity occurrence number per activity sublog size

The reported measures in Table 6.3 show good performances for the generated event log from the second phase in our approach. In fact 93% of the event log activities are significant resulting in 85% correct occurrences and 88% correct classifications in terms of speech acts. As for the obtained threads, relevant information values have contributed to reassembling emails while ensuring threads consistency in terms of instance and business context with an average measure equal to 0.85. For threads of low consistency measures, they generally results in reassembling various conversations which increases their number of emails. This explains obtaining long threads in our event log as it is shown in Figure 6.8.

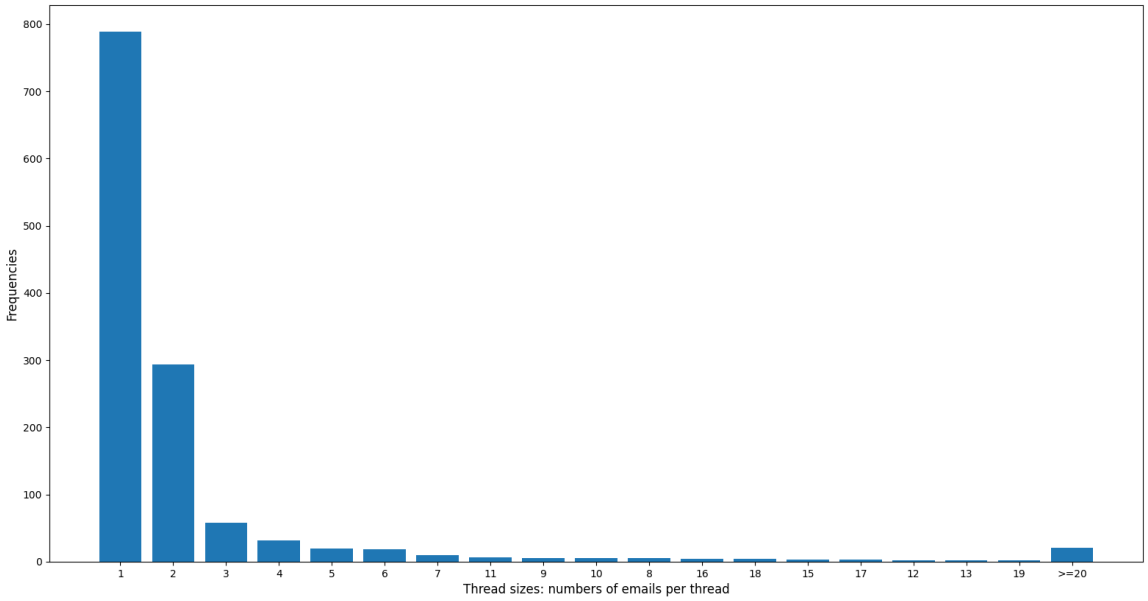


Figure 6.8: Distribution of number of threads per size (i.e. number of emails)

6.5.3.2 Precision variation per activity & employee

In what follows, we detail the variation of activity and speech acts occurrences' precision according to activities and employees. For this purpose, we report the features of the activity sublogs of the fifty most occurring ones w.r.t (i) their size, (ii) their activity and speech act occurrences' precision, and (ii) the number of the involved senders. We provide their sublogs (in form of csv files) as well as details of those of lower size in this link[2].

Table 6.4 summarizes the sublog features of these activities. Each sublog is identified by an ID (e.g. el_4), the correspondent activity (e.g. *'create_make deal{nuleric}{ticket}'*), the number of activity occurrences N_{Occ} (e.g. 113) and the number N_{sender} of the actors sending the related emails (having sending more than 5 emails). Pr_{Occ} and Pr_{SA} refer to the discovery precision of; (i) activity occurrences, and (ii) activity speech acts. They reflect their variation according to emails' senders through this format: $WF \pm sd$ where; WF is the weighted average of precision per sender and sd is their standard deviation. The obtained results show good performances in terms of precision per activity. They show also how they can be affected by; (i) activity (e.g. 0.26 difference between Pr_{Occ} of el_{16} and el_1), and (ii) email's sender (e.g. $sd = 0.22$ for Pr_{SA} of el_8) which is due to a variation in employees' writing style.

6.5.4 Experimental tool

To carry out the evaluation of our results, we have conceived a tool that analyses emails of each employee to discover his/her activities, the related occurrences and the related speech acts. In order to assess the validity of our results, we have integrated in this tool a graphical interface component to visualize them and to assist us in the annotation process. Our tool

Table 6.4: SubLogs of the most occurring activities

ID	***Activity***	N_{Occ}	Pr_{Occ}	Pr_{SA}	N_{sender}
el_1	flow deal{gas}{price}	189	0.95 ± 0.07	0.95 ± 0.04	11
el_2	change_convert deal{numeric}{price}{ticket}	118	0.98 ± 0.02	0.92 ± 0.04	5
el_3	enter deal{numeric}{ticket}{meter}	117	1.0 ± 0.0	0.9 ± 0.09	5
el_4	create deal{numeric}{ticket}	113	1.0 ± 0.0	0.93 ± 0.08	4
el_5	sell_trade numeric0mw{pricenumeric}{hour0end}{he0numeric}	99	1.0 ± 0.0	0.91 ± 0.0	1
el_6	set_put_fix_determine_adjust deal{numeric}	94	0.94 ± 0.05	0.76 ± 0.14	4
el_7	conduct interview{telephone}{informal}{eb0numeric}	93	0.98 ± 0.04	0.9 ± 0.04	2
el_8	extend deal{numeric}{rest}	90	1.0 ± 0.0	0.8 ± 0.22	5
el_9	see_determine_check_view deal{numeric}	83	0.89 ± 0.11	0.85 ± 0.1	5
el_{10}	send_attach resume{version}{electronic}	81	0.88 ± 0.03	0.94 ± 0.03	3
el_{11}	allocate volume0management{deal}{contract}{meter_mtr}	71	1.0 ± 0.0	0.78 ± 0.01	3
el_{12}	purchase_buy gas{plant}	69	0.9 ± 0.06	0.92 ± 0.01	2
el_{13}	attach spreadsheet	62	1.0 ± 0.0	1.0 ± 0.0	2
el_{14}	attach_transmit_send agreement{sheet}{discussion}{cover}	60	0.85 ± 0.24	0.97 ± 0.0	2
el_{15}	set_determine_arrange interview{phone}	60	0.82 ± 0.1	1.0 ± 0.0	2
el_{16}	make decision{deal_trade}	59	0.69 ± 0.4	0.9 ± 0.0	2
el_{17}	handle_cover deal{numeric}{meter_mtr}	58	1.0 ± 0.0	0.82 ± 0.07	3
el_{18}	show deal	57	0.82 ± 0.14	0.76 ± 0.18	4
el_{19}	forward resume{request}{associate0program}	52	0.93 ± 0.04	1.0 ± 0.0	2
el_{20}	send_attach_ship transport0contract{deal}{term}	49	0.83 ± 0.0	0.79 ± 0.0	1
el_{21}	invite_receive personnelpronoun{audience_interview}{meeting}	49	0.85 ± 0.0	0.88 ± 0.0	1
el_{22}	set_arrange_put personname{meeting}{assistant}	44	0.76 ± 0.0	0.85 ± 0.0	1
el_{23}	revise numeric{meter}{deal}{volume}	41	1.0 ± 0.0	0.95 ± 0.0	1
el_{24}	sell_trade natural0gas{plant}{counterparty}	41	1.0 ± 0.0	0.89 ± 0.0	1
el_{25}	make change	40	1.0 ± 0.0	0.82 ± 0.17	2
el_{26}	schedule locname{zone}{load}	37	1.0 ± 0.0	0.94 ± 0.0	1
el_{27}	add deal{numeric}	36	0.89 ± 0.09	0.88 ± 0.11	2
el_{28}	use_apply model_simulation{case0study}{example_model}	35	0.52 ± 0.0	0.87 ± 0.0	1
el_{29}	purchase_buy numeric0mw{pricenumeric}{hour0end}	34	1.0 ± 0.0	0.92 ± 0.0	1
el_{30}	make reservation{hotel}{dinner}	33	1.0 ± 0.0	0.79 ± 0.08	2
el_{31}	make presentation{topic}{student}{energy0derivative}	30	0.88 ± 0.0	0.9 ± 0.0	1
el_{32}	long numeric0mw{hour0end}{orgname0book}{pricenumeric}	27	0.96 ± 0.0	0.96 ± 0.0	1
el_{33}	schedule_reschedule interview{telephone}	27	0.95 ± 0.06	0.95 ± 0.03	2
el_{34}	keep_hold personnelpronoun	26	0		
el_{35}	schedule_reschedule meeting{assistant}	26	1.0 ± 0.0	0.92 ± 0.06	2
el_{36}	send_attach info_information{unit}{numeric}	26	0.96 ± 0.07	0.91 ± 0.06	2
el_{37}	wrap_roll deal{numeric}	24	1.0 ± 0.0	0.86 ± 0.12	2
el_{38}	reach cell0phone	24	1.0 ± 0.0	0.82 ± 0.11	2
el_{39}	run oomc0mean{calc}	23	0.95 ± 0.0	0.94 ± 0.0	1
el_{40}	see file	21	0.92 ± 0.07	1.0 ± 0.0	2
el_{41}	pay cost_price{balance0energy}	20	1.0 ± 0.0	0.79 ± 0.0	1
el_{42}	receive price{numeric0mw}{balance0energy}	20	0.92 ± 0.0	0.82 ± 0.0	1
el_{43}	attend meeting	20	1.0 ± 0.0	1.0 ± 0.0	2
el_{44}	bill pricenumeric{deal}	19	1.0 ± 0.0	0.89 ± 0.0	1
el_{45}	receive gas	19	0.36 ± 0.0	0.75 ± 0.0	1
el_{46}	settle_resolve issue	18	1.0 ± 0.0	1.0 ± 0.0	1
el_{47}	use_apply deal{numeric}	18	0.85 ± 0.12	0.91 ± 0.08	2
el_{48}	expire deal	18	1.0 ± 0.0	1.0 ± 0.0	2
el_{49}	deliver gas	18	0.92 ± 0.0	1.0 ± 0.0	1
el_{50}	receive_incur gas	18	0.71 ± 0.0	1.0 ± 0.0	1

is implemented using *Python* and some programming libraries to manipulate data which are mainly *Pandas* to handle the raw data and *Pyspark* to parallelize the implementation of our algorithms. It is composed of three main components:

A pre-processing component: This component ensures the first step in our approach including the application of natural language pre-processing operations on email bodies and subjects. We recall that we use to this end a set of natural language processing (NLP) libraries

including: (i) *re*[3] for detecting values of numeric types (after injecting regular expressions patterns: RegEx), (ii) *Spacy*[4] or/and *TreeTagger*[5] to detect named entities, to lemmatize words and to detect their subtrees and their syntactic, part of speech and dependency tags. This component exports the final results in the form of json files.

A back-end component: This component ensures Step 2.1, Step 2.2 and Step 2.3 in our second phase. It retrieves the pre-processed emails of each employee from the stored json files and it analyses them to discover his/her activities (including their components: activity name, business data and business information), the related occurrences and speech acts.

A front-end component: This component interacts with the user by allowing him/her to enter the email address of the employee whose emails he/she wants to analyze. Then, it visualizes the obtained results through a graphical interface. We have implemented this interface using the *Tkinter* library. Figure 6.9 shows the main blocks/windows forming it, which are:

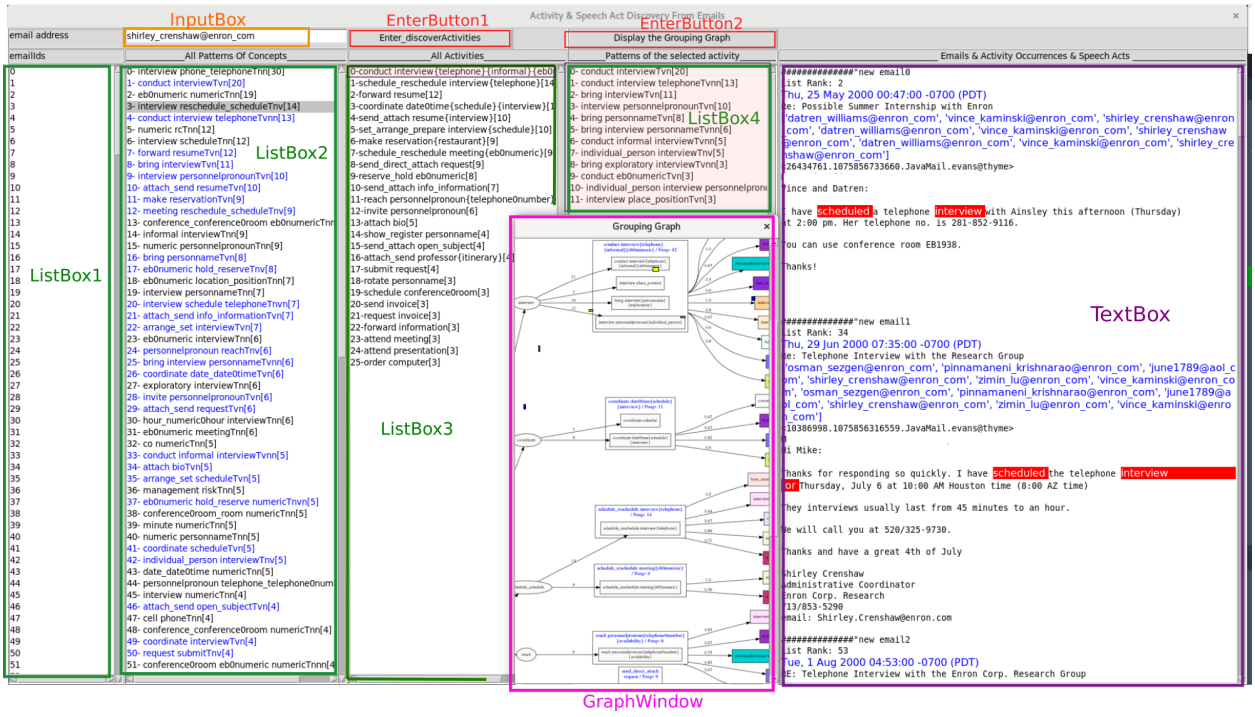


Figure 6.9: The front-end component of our experimental tool

- Entry block: it is composed of: (i) an input box (*InputBox*) where a user could type the email address of an employee, (ii) a first enter button (*EnterButton1*) that must be clicked to launch the analysis on the pre-processed emails of the selected employee, and (iii) a second enter button (*EnterButton2*) that must be clicked if the employee wants

[3]https://docs.python.org/3/library/re.html
[4]https://spacy.io/
[5]https://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/

to visualize the organization of patterns grouped per inclusion by verb concepts and by activity;

- Email visualization block (*TextBox*): it visualizes emails based on filtering criteria related to their IDs, the pattern or the activity that it contains (more details in what follows);

- Email IDs block (*ListBox1*): This block lists the set of IDs attributed by our algorithm to the analysed emails. By clicking on one ID, the related email will be displayed in the email visualization block (*TextBox*). Figure 6.10 shows two examples of the displayed contents in case of clicking on two email IDs (in relation to two different employees). We illustrate with the first example (Example 1) the different displayed parts. The first part ($P1$) shows the list of the discovered activities in the email, their related speech acts (SA) and actor indications (ActInd). Each one of these activities is highlighted by a color reflecting its speech act. In our case, we adopt: (i) the green color to refer to the information speech act, (ii) the red color to refer to the intention speech act, (iii) the pink color to refer to the request speech act, and (iv) the purple color to refer to the request information speech act. The second part ($P2$) displays the set of email related attributes (i.e. recipient, subject and email real ID). The third part ($P3$) displays the email main body. It highlights the words inducing the detection of the activity patterns with colors reflecting their speech acts. This is useful in the annotation process for validating activities, their related occurrences and speech acts. In fact, it helps, especially for longer emails, in locating activities in them, which speeds up the process of searching these activities in the email textual contents. We show in Figure 6.10 a second example (Example 2) of a longer email displayed by our tool when clicking on another ID and which contains more activities (i.e. 5). The final part in our email visualization block ($P4$) shows the email conversation history, which provides a better understanding of the business context of the discovered activities;

- Patterns Block (*ListBox2*): This block lists all patterns discovered from the analysed emails. These patterns are displayed in ascending order according to the number of emails where they appeared (this number is indicated between braces after each pattern label). By clicking on one pattern label (e.g. *'3- interview reschedule_schedule' [14]* in Figure 6.9), all emails containing it will be displayed in the email visualization block (*TextBox*). Words inducing the detection of such pattern are highlighted in red color to facilitate locating it in the emails for validation/annotation purposes (e.g. see the displayed emails in the *TextBox* of Figure 6.9);

- Activity Block (*ListBox3*): This block lists all the activities discovered from the analysed emails. These activities are displayed in ascending order according to the number of emails where they appeared. By clicking on one activity label (e.g. *conduct interview{telephone}{informal}{eb0numeric}*), all emails containing it will be displayed in the email visualization block (*TextBox*) while locating the occurrences of their related patterns. Additionally, all patterns characterizing its activity name component will be displayed in the activity patterns block (*ListBox4*);

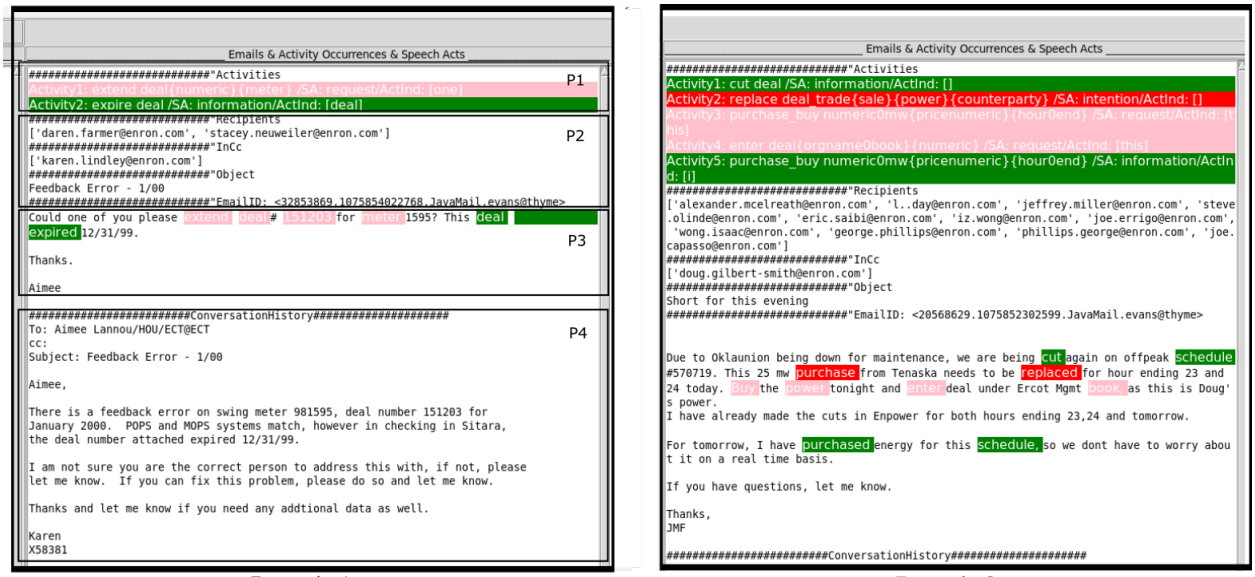


Figure 6.10: Email visualization block when clicking on an email ID

- Graph window (*GraphWindow*): This window is displayed by clicking on the second enter button (*EnterButton2*). It displays a graph resuming the organization of patterns by verb concepts and by activity (after grouping them by inclusion). Such graph is generated using the *pydot* library. It hierarchically organizes activities' patterns in the form of tree (i.e. see the two parts of the activity patterns tree obtained after analysing emails of the employee E_4 in Figure 6.11 and Figure 6.12). We provide in this link[2] (See the third section in such link related to discovering activities) the visualizations results that we have obtained with the other employees considered in our experiments. The parent nodes in the generated graphs present the different verb concepts appearing in the discovered activities (e.g. *'hold_reserve'* and *'interview'* in Figure 6.11). Each rectangular child node refers to a group label of included patterns and it is linked to the verb concept's node to which it was assigned. Child nodes belonging to the same activity (e.g. the nodes linked to the parent node *'interview'*: *'conduct interview{telephone}{informal}{eb0numeric}'* and *'interview{personnelpronoun}{individual_person}'*) are grouped through a rectangle. At the top of this rectangle, the representative activity name and the number of emails containing it are indicated. The activity patterns tree additionally shows the set of business information that are linked to each activity (see the colored nodes in Figure 6.11 and Figure 6.12) as well as the related coexistence coefficients (see the labels of the arrows linking activities to their business information nodes).
  This activity patterns tree has the advantage of outlining, for each employee, his/her overall activities and the related activity name and business information patterns. It helps in: (i) visually assessing the grouping quality of patterns into activities (e.g. by identifying if there are child nodes that are over or under grouped), (ii) easily identifying the business information that have induced the regrouping of a set of child nodes. Taking the example of the parent node *'interview'* in our tree example (Figure 6.11), it shows the ability of our algorithm to group patterns expressing differently the same activity

(i.e. interview a candidate) while sharing: (i) rephrasing relations (e.g. conduct/bring interview and interview a person), and (ii) similar business context (e.g. 'interview', 'position' are not synonyms but the patterns that they form share the same business context). It also shows its ability in separating child nodes that do not share the same business context, e.g. the child nodes linked to the verb concept *'hold_reserve'* (i.e. *'reserve_hold eb0numeric'* and *'make reservation{restaurant}'*), those linked to the verb concept *'attach_send'* (e.g. *send_attach request* and *send_attach resume{interview}*) and those linked to the verb concept *'reschedule_schedule'* (i.e. *schedule_reschedule interview{telephone}* and *schedule_reschedule meeting{eb0numeric}*).

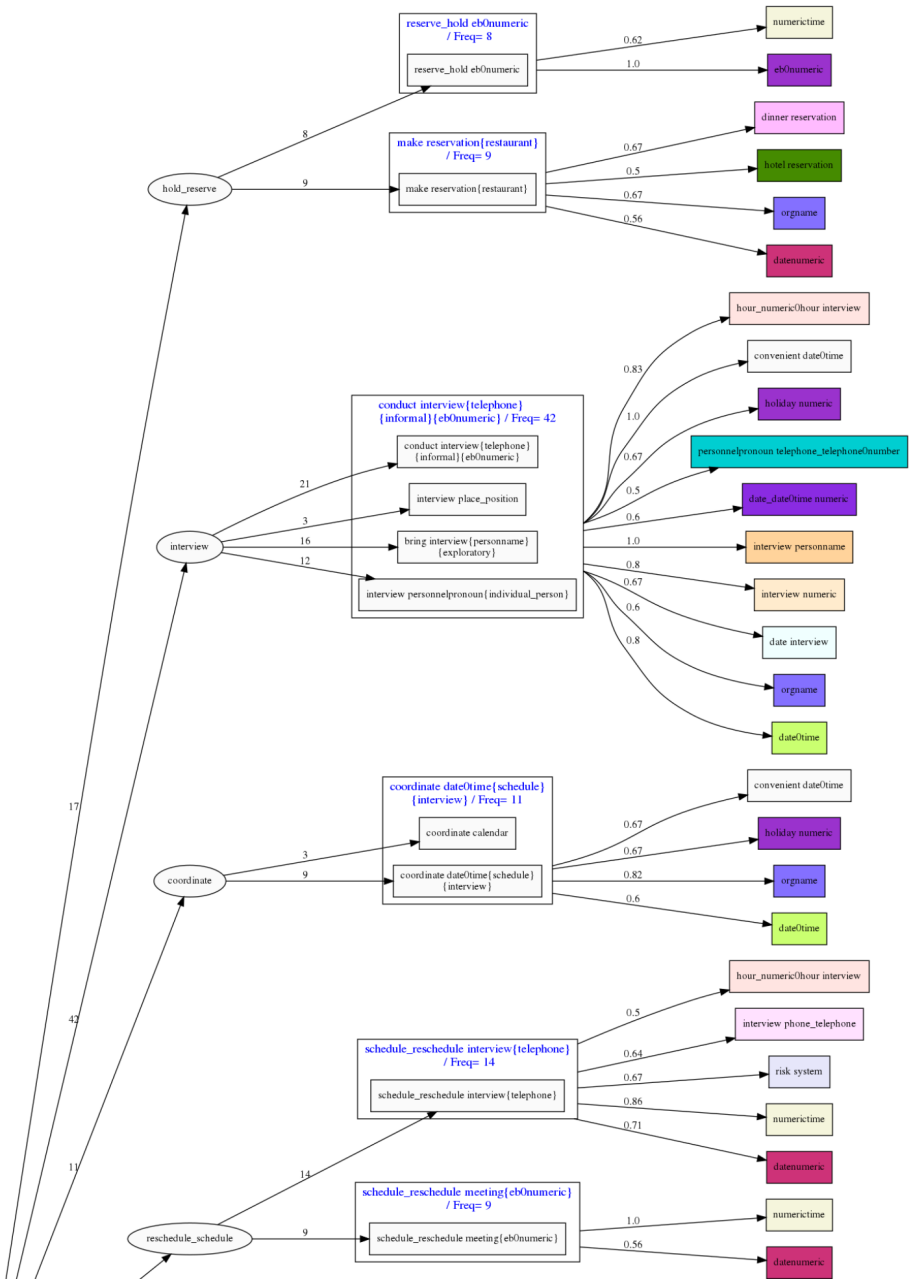


Figure 6.11: Activity patterns tree after analysing emails of the employee E4: Part1

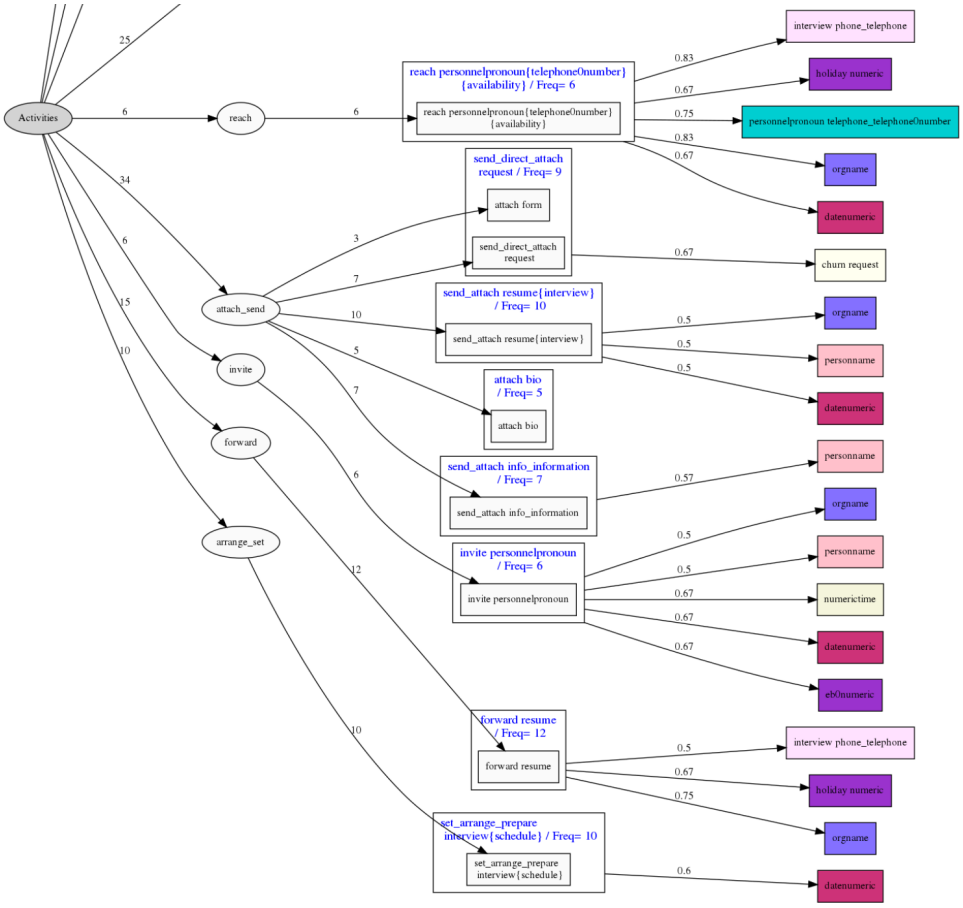


Figure 6.12: Activity patterns tree after analysing emails of the employee E4: Part2

6.6 Conclusion

In this chapter, we achieved the first part of our second objective (**Objective 2.1:** Automate the generation of a structured event log from emails) and we answered the second question (**Q2**) raised in the thesis problematic (Section 1.2.2), which is : How to discover BP knowledge from emails? according to the following sub-questions:

- Q2-2: How to select, from emails, relevant information in relation with BP elements ?
- Q2-3: How to discover BP elements considering the different degree of their granularity expression ?
- Q2-4: How to discover BP elements considering their multiplicity in emails ?

To this end, we were mainly based on the characterisation of the discovered activities as well as their components in terms of patterns, which were obtained from the previous chapter (Chapter 5). We used such characterization for discovering activity occurrences in emails that are not necessary considered in the learning part. In this way, we were able to discover multiple occurrences of the same or of different activity components in one email. Additionally, we were able to locate each one of them in the corresponding email to extract further information related to their speech acts, actors' contribution and the real values of their business data. Afterwards, we used such real values as well as the email receivers addresses to select relevant information approximating the instance identifier that are mandatory for threads' construction.

For evaluating our proposals, we have been mainly based on emails retrieved from the public dataset Enron. We carried out various experiments to show the effectiveness of our results and we have publicly provided the obtained results.

We agree that some shortcomings could be derived at three levels.

- At speech act discovery level: We admit that further analysis (on larger set of employees) must be carried out to study if our main assumptions remain valid while varying employee writing styles. These assumptions concern mainly our rules used for speech act discovery and the relevance of the selected features.
- At thread construction level: For selecting each relevant information value approximating an instance identifier, we relied on a single value of a single information type. Actually, an approximation through a combination of values related to different information types must be studied, instead of one. Additionally, an instance could be equally expressed (as the case of activity names) through a combination of words (that do not necessary include numeric or named entity tags) such as the case of research project names in the context of project management BP fragments;
- At evaluation level: We have been mainly based on calculating the discovery precision of the speech acts and the activity occurrences included in the generated event log. The

recall was only studied at the level of the speech acts of a set of activity occurrence samples of three employees. The thread construction step was additionally evaluated on the basis of a consistency metric that does not assess the recall of grouping emails into threads. Actually, the instance information is the BP element with a low degree of granularity comparing to the overall considered BP elements in our work. This means that if the BP activities would be frequent, the instance information values would probably tend to be unique. In other words, at the level of the execution traces of the same BP, there would be a much greater variety of instance values than at the level of activity types. This actually makes the instance annotation phase not always trivial, especially if we start with thousands of emails (which is our case) belonging to a set of employees and whose content we do not dispose a priori knowledge. As an alternative, we tried to evaluate the consistency aspect in the thread construction part. This assesses to what extend the constructed threads contain emails that handle the same BP fragment instance;

In the next chapter, we will focus on detailing our algorithmic approach that we have introduced to mine the generated event logs (Chapter 7) in order to automate the discovery of BP w.r.t. their multiple perspectives.

Chapter 7

Event Log Mining

Contents

7.1 Introduction

This chapter focuses on the third phase in our overall approach that mines the generated event log (at the level of Chapter 6) to discover BP fragments. Existing approaches that discovered BP from emails were most often limited to the discovery of the functional and behavioral perspectives [46, 24, 22, 55, 11, 52]. The behavioral perspective was generally discovered in the form of activity models (i.e. activity control flow) with imperative description (at the

exception of one work [24] that adopted declarative description). Nevertheless, such activity models are compatible with highly structured BP which is not the case of those performed through emails. Additionally, they generally rely on existing algorithms (e.g. [96, 37, 3, 1]) that requires explicit timestamp information of events for discovering BP models. This is actually limited as the timestamps referring to the date of the occurrences of the events are most often absent in emails.
For other BP perspectives, only few studies have studied the data and the organizational perspectives [86, 48]. However, several aspects were not largely and concretely studied to date: For the organizational perspective, this includes the discovery of actors contributions in performing activities (e.g. execution, request, planning, etc). As for the data perspective, this includes the discovery of the informational entities manipulated by BP activities. For instance, previous studies [86, 48] were limited to detecting some pre-defined data types such as the attached documents and email-included web pages description.

In this chapter, we propose an event log mining phase that discovers BP fragments w.r.t multiple perspectives. We recall that we have defined four perspectives to be discovered for each BP fragment in this phase:

1. Functional perspective: It describes the composition of a BP of atomic business goals referring to the set of activities to be performed in its context;
2. Data perspective: It describes the set of artifacts manipulated by BP fragment activities and the relationships between them;
3. Organizational perspective: It describes the set of actors and actor groups involved in performing each activity, their related contributions and how they interact between each others in order to perform each activity;
4. Behavioral perspective: It describes the sequential constraints between event types;

As illustrated in Figure 7.1, this phase is composed of five main parts. It first analyses event log in order to discover activity artifacts from their corresponding business information (i.e. Part 3.1). Then, it groups BP elements (i.e. activities, actors & artifacts) into BP fragments (Part 3.2). In this way, the functional perspectives of BP fragments are directly deduced from the obtained groups. Therefore, each BP fragment is discovered according to its data (i.e. Part 3.3), organizational (i.e. Part 3.4) and behavioral (i.e. Part 3.5) perspectives.

The first main requirement that needs to be fulfilled by our proposed solutions at this level is to allow regrouping business information (BP elements) into artifacts (BP fragments): (i) without disposing a priori knowledge about them, and (ii) while assigning one business information (BP element) to multiple groups referring to different artifacts (BP fragments) as it could appear in the context of more than one artifact (BP fragment). To this end, we propose two overlapping clustering algorithms to ensure artifact and BP fragment discovery.

An overlapping clustering is defined as a non-exclusive grouping of information (i.e. one information could belong to multiple groups). A review established in the context of over-

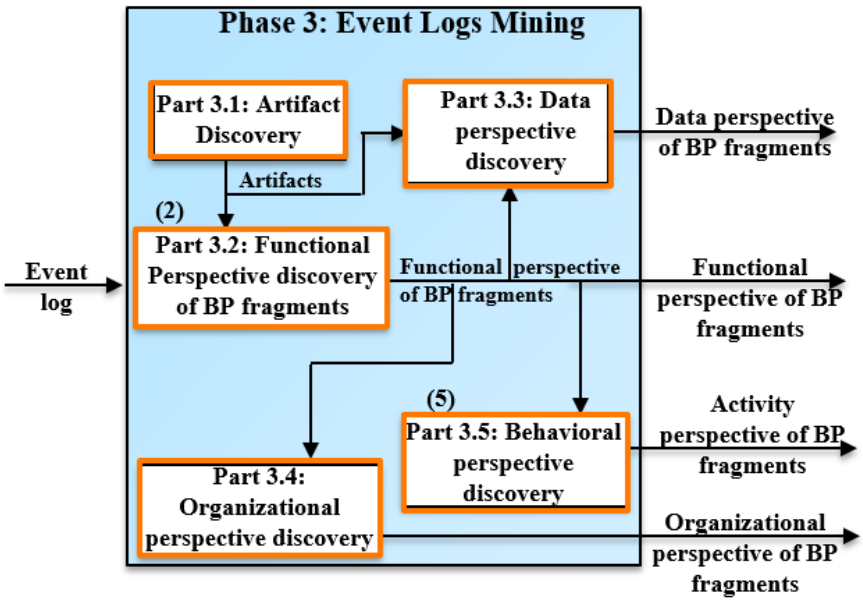


Figure 7.1: Framework extract: Main parts in the event log mining phase

lapping clustering algorithms [10] have reported some recent ones that could be divided into two types according to their methodology:

1. Overlapping graph-based methods (e.g. OClustR [74]): information in these methods are represented in the form of directed or undirected graph. Then clusters are built into two phases; an initialization phase where a set of initial clusters are built from the information graph, and an improvement phase where further operations are applied on the initial clusters to reduce both the number of clusters and their overlapping;

2. K-means extension methods (e.g. fuzzy K-means [42]): They are based on associating a numerical representation (e.g. vector) to each information. The common methodology adopted by these methods is to find representative centers of each cluster in a way that for a given data vector it can determine where this vector belongs. This can be implemented by measuring a similarity metric (e.g. euclidean, cosine) between the input vector and the data centers. Data allocations to clusters can be more refined by a user specified threshold [17] or an automatically computed maximum distance [9]. However, such methods always require users to enter or estimate the number of clusters

In this work, we were inspired by the first methodology type when partitioning information into groups that overlap to avoid defining the number of groups in advance (as it is required in the second methodology). Our methodology for overlapping clustering information has the following features:

- It injects the target knowledge (artifacts / BP fragments) characterizations in the initialization phase so that a set of initial groups is automatically built.

- It further refines information allocation to the obtained groups: additional information

could be allocated to each group if some membership criteria (defined according to the target knowledge characterizations) are verified.

- It does not need to set or estimate the number of groups. However, it requires a predefined threshold to implement group membership criteria.

The second main requirement that needs to be fulfilled is to take into account email specifities when mining sequencing constraints and artifact relations from the our event log. We recall that these specifities results in generating an event log which lacks of precise information concerning event timestamps and instantiating artifacts/business data and BP fragments. To tackle such information shortage, we propose to approximate events sequencing from emails and threads rather than relying on existing discovery algorithms requiring precise information referring to event timestamps. Additionally, we propose to approximate artifact cardinalities from the occurrences of their attributes in threads.

In the following, we start by presenting our solutions for artifact discovery (Section 7.2) and for BP fragment discovery (Section 7.3). Afterwards, we introduce our approaches for discovering the data (Section 7.4), organizational (Section 7.5) and behavioral (Section 7.6) perspectives. Finally, we validate our approach for event logs mining and we interpret our experiment results (Section 7.7)

The work in this chapter was submitted in the Information Systems journal [33].

7.2 Artifact discovery

We remind that artifacts refer to the informational entities used or generated by BP fragment activities. Such entities are primordial for ensuring the discovery of BP fragments as well as their data perspectives. This section focuses on the first part in the event log mining phase aiming for discovering artifacts. In this part, we basically regroup activities in the way that each obtained group is formed by activities manipulating the same artifact. For this purpose, we rely on an overlapping clustering technique to regroup activities in terms of business information similarity. We assume that: (i) each activity group is formed by activities sharing the same set of business information, and (ii) the common business information shared by activities belonging to the same group forms an artifact. The use of an overlapping clustering technique ensures that one activity could be assigned to more than one group (i.e. artifact).

This part defines one activity group as a set of activities having pairwise similarity according to the same set of business information (Definition 7.2). This means that all activity pairs belonging to the same group are similar according to the same business information features. Once obtaining such groups of activities, one artifact will be associated to each group and will be built from the set of business information that have induced its construction.

As summarized in Figure 7.2, the artifact discovery part is composed of two steps that we further detail in what follows: (i) Overlapping clustering of activities by business information

similarity, and (ii) Build artifacts from the business information of the activities of each group.

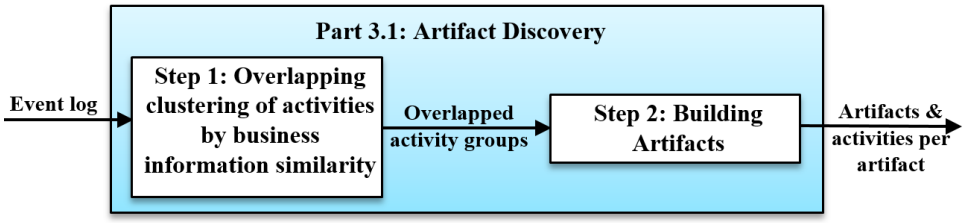


Figure 7.2: Artifact discovery main steps

7.2.1 Step 1: Overlapping clustering of activities by business information similarity

To group activities, two notions must be determined: (1) pairwise similarity between activities (Definition 7.1), and (2) an activity group forming an artifact (Definition 7.2).
For the pairwise similarity between activities, we extend our definition for pairwise similarity between action patterns (Definition 5.7) as follows:

Definition 7.1. ***(Activity Pairwise Similarity)*** *Let consider the following notations:*

- Act_1 *and* Act_2 *be two activities* $\in \mathcal{A}$*;*
- $BI_i \subset \mathcal{BI}$ *is the set of business information of the activity* Act_i*, where* $i \in [1,2]$*;*
- $BD_i \subset \mathcal{BD}$ *is the set of business data of the activity* Act_i*, where* $i \in [1,2]$*;*
- $f_{actBC} : BI^* \rightarrow \mathcal{W}$ *is the business context function that returns, from a set of business information, the set of business context words forming them and that are different from numeric and named entities' tags;*
- $f_{W_{id}} : \mathcal{W} \rightarrow \mathcal{ID}$ *be the function that returns for a word* ($\in \mathcal{W}$)*, the set of email IDs where it appears;*
- $f_{act_{id}} : \mathcal{A} \rightarrow \mathcal{ID}^*$ *be the function that returns the set of email IDs where an activity* $Act \in \mathcal{A}$ *appears*
- $f_{act_{coexist}} : \mathcal{A} \times \mathcal{W}^* \rightarrow \mathbb{R} \cap [0,1]$ *be the function that measures how much an activity (Act) coexists with a set of business context words* ($B \subset \mathcal{W}^*$) *such that* $f_{act_{coexist}}(Act, B) = \frac{|\bigcup_{b \in B} f_{W_{id}}(b) \cap f_{act_{id}}(p_{act})|}{|f_{act_{id}}(Act)|}$*;*

The pairwise similarity between two activities Act_1 *and* Act_2 *is defined as a tuple* $Sim_{act_{pairwise}} = (Sim_{act}(Act_1, Act_2), BC_{inter}, BD_{inter})$ *where:*

- $BC_{inter} = f_{actBC}(BI_1) \cap f_{actBC}(BI_2)$;
- $BD_{inter} = \{b,\ b \in BD_1 \cup BD_2 \mid f_{actBC}(\{b\}) \cap BC_{inter} \neq \emptyset\}$
- $Sim_{act} : \mathcal{A} \times \mathcal{A} \rightarrow [0,1]$ *is the similarity measure function defined as follows:*

$$Sim_{act}(Act_1, Act_2) = min(f_{act_{coexist}}(Act_1, BC_{inter}), f_{act_{coexist}}(Act_2, BC_{inter}));$$

The pairwise similarity between activities is defined according to three elements: (i) the set of business context words that are shared between them (BC_{inter}), (ii) the set of business data of the two activities whose words include one of the business context words found in common (BD_{inter}), and (iii) a similarity measure ($Sim_{act}(act_1, act_2)$) that reflects the minimal degree of coexistence of such set of business context words with both activities. With such measure, we guarantee (as in the case of action patterns pairwise similarity) that the two activities will have a high similarity measure if and only if both of them have high coexistence coefficient with the set of business context words that they share. This is useful for building groups of activities having the same degree of pairwise similarity and according to the same business context features. Based on such notion of activity pairwise similarity, we extend Definition 5.8 to define an activity group forming an artifact as follows:

Definition 7.2. ***(Artifact Activity Group)*** *Let consider the following notations:*

- $f_{BC} : \mathcal{A} \rightarrow \mathcal{W} \cap \mathcal{BC}$ *be the function that returns for each activity* $Act \in \mathcal{A}$*, the set of its related business context words;*
- $f_{act_{coexist}}$ *be the function that measures how much an activity coexists with a set of business context words as defined in Definition 7.1;*
- α *and* α' *two user parameters such that* $\alpha \in \mathbb{R} \cap]0,1]$ *and* $\alpha' \in \mathbb{R} \cap]0,1]$
- $Ar \in \mathcal{AR}$ *be an artifact*

$G_{Ar} \subset \mathcal{A}$ *is considered as a group of activities manipulating the same artifact* Ar *such that* $B_{G_{Ar}} = \bigcap_{Act \in G_{Ar}} f_{BC}(Act)$ *and* $\forall Act_i, Act_j \in G_{Ar} \times G_{Ar}$*:*

(i) $Sim_{act}(Act_i, Act_j) \geq \alpha$; *AND*

(ii) $f_{act_{coexist}}(Act_i, B_{G_{Ar}}) \geq \alpha'$; *AND*

(iii) $f_{act_{coexist}}(Act_j, B_{G_{Ar}}) \geq \alpha'$;

An activity group forming an artifact must be composed of a set of activities that have the same degree of pairwise similarity and according to the same business context features (which is guaranteed by the scalar $\alpha \in]0,1]$). To perform an overlapping grouping of activities with such requirements, we first apply our algorithm (Algorithm 3) introduced in the previous chapter (Chapter 6) where we adapt the inputs as follows:

- We provide the set of activities to be clustered instead of the set of patterns ($Patterns$);
- We provide the similarity matrix resuming pairwise similarity between activities for the input M_{sim};
- We provide the pairwise intersection between activities in terms of business context words for the input $dicPairs_{InterBC}$;
- We provide the α value for the input Th_s;

The goal of such algorithm is to obtain initial activity groups. These groups are then improved by removing redundant ones merging those of inclusion relationships (but without assigning activities that would belong to more than one group to the most frequent one as we proceeded previously when grouping action patterns). Afterwards, each one will be represented by the set of business context words in common with all its activities. Finally, additional activities will be allocated at each group if they have high coexistence coefficient (according to the α threshold) with its representative business information.

7.2.2 Step 2: Building Artifacts

This step builds artifacts from the business information related to each activity group obtained in the first step. This is mainly about identifying artifact components which are : (i) artifact name, and (ii) artifact attributes. For each activity group, we infer the artifact name from the concatenation of business context words characterizing it. As for the artifact attributes, we first select business data of the activities forming it. Then, we keep those that contain its business context words. Finally, we infer attributes from the remained business data after removing redundancy and handling included ones.

7.3 Functional perspective discovery: BP fragments discovery

At the level of this second part, we dispose the following BP elements: (i) activities, (ii) artifacts and (iii) actors that correspond to the interlocutors (i.e. senders & receivers) of activity related emails. The goal is to regroup these BP elements into BP fragments. As one BP element could belong to multiple BP fragments, we equally apply an overlapping grouping algorithm. As illustrated in Figure 7.3, this algorithm is mainly based on two main steps that we further detail in what follows: (i) Obtain initial groups of BP elements where we intent from each one to form a starting point towards a BP fragment construction (Section 7.3.1), and (ii) assign additional BP elements to the initial groups (Section 7.3.2)

In the following, let $\mathcal{BE}$ denote the set of BP elements of the following types: activity, artifact and actor.

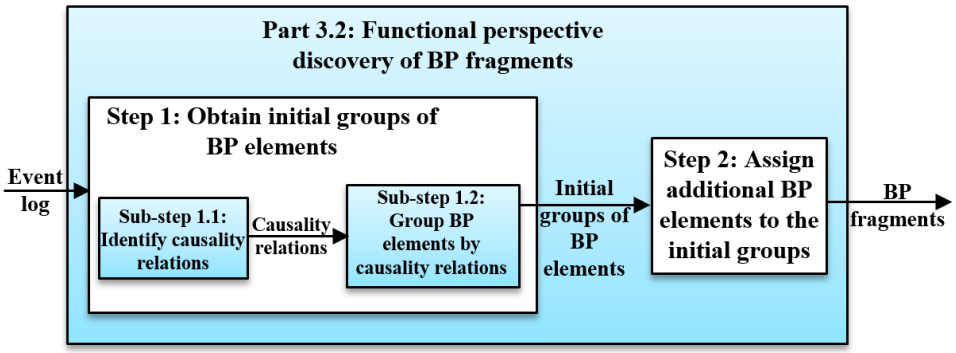


Figure 7.3: BP fragment discovery: Main steps

7.3.1 Step 1: Obtain initial groups of BP elements

This step analyses the occurrences of activities, actors and artifacts in the event log provided as input. It aims to infer from them the sets of BP elements that form initial groups towards BP fragments construction. One of the important point to define is what makes one BP fragment different from another. This is useful for obtaining a kind of characterization (in terms of activities, artifacts and actors) that uniquely differentiates it from another. In the case of BP fragments executed through emails, we suppose that a combination of ***causality relations*** (Definition 7.3) between activities, actors and artifacts could differentiate one BP fragment from another. Based on such assumption, we first identify causality relations by uncovering association rules of high confidence between BP elements from emails threads (i.e. Sub-step 1.1 in Figure 7.3). Then, we group BP elements on the basis of the identified relations (i.e. Sub-step 1.2 in Figure 7.3). Finally, we associate a BP fragment to each obtained information group.

Sub-step 1.1: Identify causality relations: We define a causality relation as follows (Definition 7.3):

Definition 7.3. ***(Causality Relation)*** *Let $f_{id} : \mathcal{BE} \rightarrow \mathcal{ID}$ be the function that returns the set of thread IDs where a BP element ($\in \mathcal{BE}$) appears. Let X and Y be two BP elements $\in \mathcal{BE}$. Let $\gamma \in [0, 1]$ be a pre-defined threshold. X and Y have a causality relation if $Pr(Y|X) \geq \gamma$ or $Pr(X|Y) \geq \gamma$, where: $Pr(X|Y) = \frac{|f_{id}(X) \cap f_{id}(Y)|}{|f_{id}(Y)|}$. X and Y have a causality relation with direction $X \rightarrow Y$ if $Pr(Y|X) \geq \gamma$.*

A causality relation in our context refers to a dependency relation between two BP elements in terms of coexistence in the same emails threads. If a BP element X has a causality relation with another BP element Y (with direction $X \rightarrow Y$), this means that its occurrence in one email thread induces the occurrence of Y with a high conditional probability $Pr(Y|X) \geq \gamma$ where: $\gamma \in]0, 1]$ is a user parameter that specifies from which value, a probability is considered to be high.

To identify causality relations between BP elements, we first merge threads having at least one email in common to be considered as one thread. This enables avoiding the appearance of fake causality relations. Then, we identify four causality relations types:

- Activity-Activity causality relations: They refer to Activity - Activity association rules with high confidence measure mined from email threads. Each email thread is associated here to a record containing the list of activities appearing in it.
- Actor-Actor causality relations: They refer to Actor - Actor association rules with high confidence measure mined from email threads. Each email thread is associated here to a record containing the list of actors (i.e. interlocutors) appearing in it.
- Activity-Actor causality relations: They refer to Activity-Actor association rules with high confidence measure (according to the γ threshold) mined from email threads. Each email thread is associated here to a record containing the list of activities and actors (i.e. interlocutors) appearing in it.
- Activity-Artifact causality relation: They are directly deduced from the previous step where activities were mapped to their artifacts., i.e. if an activity is mapped to an artifact, it will have a causality relation with it.

Sub-step 1.2: Group elements by causality relations: Once identifying BP elements with causality relations, this sub-step regroups them into BP fragments in three iterations:
- `Iter1: Finding actors groups`: In the first iteration, a graph is constructed with actors having causality relations. Actors groups are identified by detecting ***weakly connected components*** [1] in the obtained graph.
- `Iter2: Finding activity groups`: In the second iteration, a graph is constructed in the way that each edge relates a couple of activities having causality relations and introduced by at least the same actors group. Activity groups will be then identified by detecting weakly connected components from the obtained activity graph.
- `Iter3: Finding BP fragments`: Relying only on coexistence relations between activities to form BP fragments would be not sufficient. This is due to the approximations made for generating BP traces (i.e. threads) or to not being able to group emails handling the same instance in the same thread (e.g. due to the absence of instance identifier in them). We consider that activities belonging to the same BP fragment would be additionally characterized by common business context and common actors. That's why, activity groups are merged in this iteration if they share the same business context and actors. Two activity groups Gr_1 and Gr_2 will be merged if they have a number $N_{actPairs}$ of activity pairs sharing the same actors and artifacts (even in the absence of causality relations between them). Each pair is composed of an activity belonging to Gr_1 and another belonging to Gr_2 while both of them have a causality relation with at least (1) one common actor, and (2) one common artifact.

7.3.2 Step 2: Assign additional BP elements to the initial groups

In this second step, each BP fragment BP_F is represented by the BP elements composing its causality relations. The goal is to assign, to each BP fragment BP_F, the remained related

[1] Given a directed graph, a weakly connected component (WCC) is a subgraph of the original graph where all vertices are connected to each other by some path, ignoring the direction of edges. In case of an undirected graph, a weakly connected component is also a strongly connected component

activities ($\subset \mathcal{A}$) that could be shared with other BP fragments. Actually, these activities could appear in different BP fragments. Consequently, their coexistence degree with one or a set of activities characterizing a given BP fragment would be logically lower than the required coexistence degree (i.e. γ) for forming causality relations of the same BP fragment.
An activity Act_1 will be allocated to an existing BP fragment BP_F, if it coexists with its activity members or at least with one activity member with a coexistence degree lower than γ. This degree is identified by a parameter $\beta < \gamma$ while; $\frac{|(\bigcup_{act \in BP_F} f_{id}(act)) \cap f_{id}(Act_1)|}{min(|\bigcup_{act \in BP_F} f_{id}(act))|, |f_{id}(Act_1)|)} \geq \beta$.

7.4 Data perspective discovery

In the third part of the event log mining phase, we focus on discovering the data perspective of the obtained BP fragments. As we previously described it (Definition 7.4), the data perspective of a BP fragment is defined according to three notions: (1) Artifact, (2) Association (of artifact-artifact or artifact-activity types) and (3) Artifact cardinality. In previous sections, BP fragments, artifacts and artifact-activity associations have been discovered. Artifact-activity associations are to be directly deduced from activities mapped to each artifact (obtained in Section 7.2) as the case of activity-artifact causality relations. In this part, artifact-artifact associations and artifact cardinalities remain to be discovered.

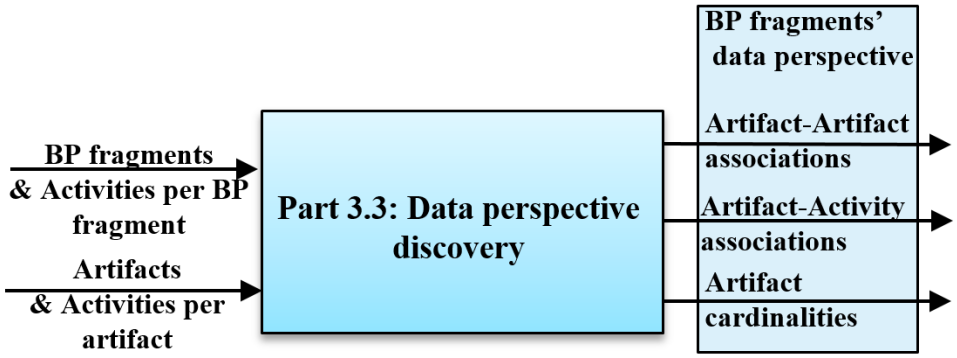


Figure 7.4: Data perspective discovery: Main Inputs & Output

We resume in Figure 7.4 the main inputs/outputs of the data perspective discovery part. For a given BP fragment, we first project the obtained event logs on its activities to obtain a BP fragment sublog. This means that we retrieve events related to the BP fragment activities. Then, for identifying **artifact-artifact associations**, we mine causality relations (Definition 7.3) between artifacts from the BP fragment sublog. Finally, we mine artifact cardinalities by estimating, for each pair of artifacts (having an association relation), the average multiplicity or number of instantiation of one artifact induced by the instantiation of the other across threads.

To formally define how we obtain such multiplicities. Let consider the following notations:

- Ar_1 and Ar_2 be two artifacts;
- $card = (m_{12}, m_{21})$ be the association cardinality between Ar_1 and Ar_2 of two multiplicities m_{12} and m_{21};
- $f_{artInst} : \mathcal{AR} \times \mathcal{TH} \rightarrow \mathcal{O}^*$ be the function that returns instances ($\in \mathcal{O}^*$) of one artifact ($\in \mathcal{AR}$) in one email thread ($\in \mathcal{TH}$);
- $Round : \mathbb{R} \rightarrow \mathbb{N}$ be the function that returns the value of a number rounded to the nearest integer;

The multiplicity m_{ij} (denoting the number of instances of Ar_i that can occur on the instantiation of Ar_j) is estimated based on the average multiplicity function $Mult_{AVG} : \mathcal{AR} \times \mathcal{TH} \rightarrow \mathbb{R}$ such that:

$$Mult_{AVG}(Ar_i, Ar_j) = \frac{\sum_k^D \frac{card(f_{artInst}(Ar_j, thread_k))}{card(f_{artInst}(Ar_i, thread_k))}}{D} \tag{7.1}$$

Where $D = Card(\{thread \in \mathcal{TH} | f_{artInst}(Ar_i, thread) \neq \emptyset \ AND \ f_{artInst}(Ar_j, thread) \neq \emptyset\})$

Multiplicities ($\in Mult$) are inferred from average multiplicity values according to these rules:

- if $Round(Mult_{AVG}(Ar_i, Ar_j)) > 1$ then $m_{ij} = 1..*$
- if $Round(Mult_{AVG}(Ar_i, Ar_j)) == 0$ then $m_{ij} = 0..1$
- if $Round(Mult_{AVG}(Ar_i, Ar_j)) == 1$ then $m_{ij} = 1$

7.5 Organizational perspective discovery

The fourth part in the event log mining phase focuses on discovering the organizational perspective of the obtained BP fragments (from the second part). For each BP fragment activity, we remind that we define such perspective (Definition 4.9) according to four notions: (i) activity actors, (ii) activity actors' groups, (iii) activity sender-receivers groups and (iv) the distribution of contributions per actor. We resume in Figure 7.5 the main steps that we apply for discovering it. We first project the obtained event log on each BP fragment activity to obtain its related sublog (Step 1, Figure 7.5). Then, we apply two types of analysis; the first one (Step 2, Figure 7.5) aims to identify activity actors and the different corresponding groups (i.e. sender-receiver groups and actor groups). The second one (Step 3, Figure 7.5) aims to infer actor contributions from event speech acts. We detail in the following sections the second and the third steps.

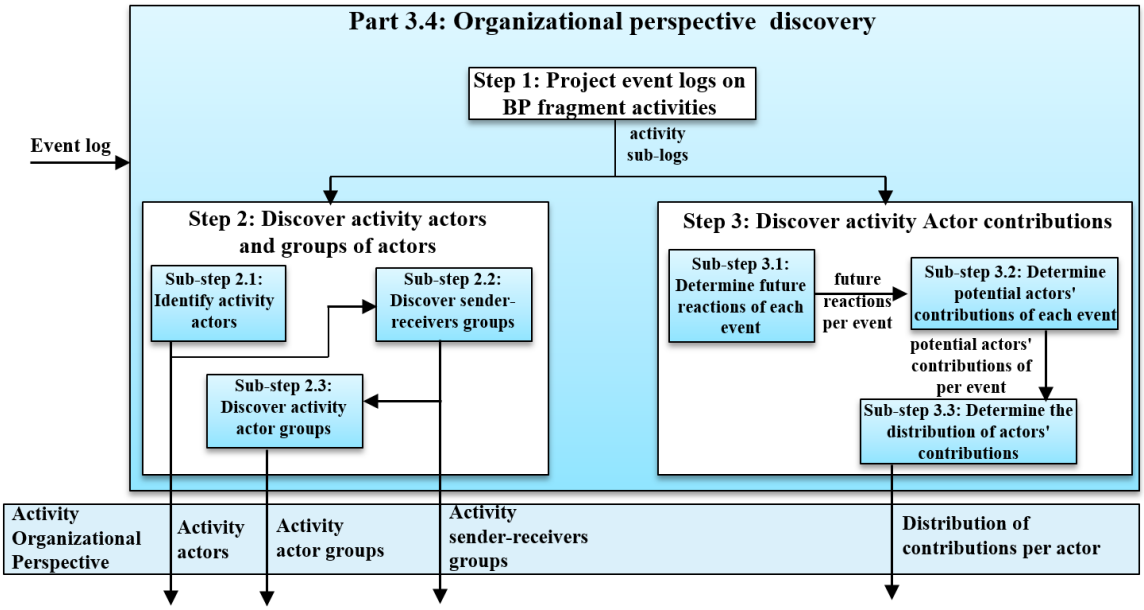


Figure 7.5: Organizational perspective discovery: Main steps

7.5.1 Step 2: Discover activity actors, actor groups and sender-receivers groups

In this step (see Step 2, Figure 7.5), we first identify activity actors (Sub-step 2.1). Then, we discover the sender-receivers groups that they form (Sub-step 2.2). Finally, we identify the global actor groups including them.

Sub-Step 2.1: Identify activity actors: For a given activity, this sub-step retrieves all the interlocutors (i.e. senders & receivers) present in its related sublog to identify its actors. These actors are then sorted in descending order w.r.t. their number of occurrences in the sublog's events to identify the most important ones. We differentiate between two types of actors:

(i) Senders: They correspond to the actors who frequently send emails about the corresponding activity. They do not have to be only senders, which means that they can also be receivers of emails related to the activity. To identify these actors, we rely on a threshold defining the minimum number of emails sent by them. This type of actors must be identified primarily to be able to discover the sender-receivers group that they form. We require that a sender frequently sends activity-related emails to be able to identify the groups of recipients that frequently receive their emails;

(ii) OnlyReceivers: They correspond to actors that receive emails about the corresponding activity. These actors may react to the emails sent by senders by replying or forwarding emails but do never send emails containing the activity;

Sub-Step 2.2: Discover sender-receivers groups: For identifying sender-receivers groups, this sub-step selects the set of senders from activity actors. Then, for each sender, it identifies the emails sent by him/her. It associates each email to a record containing its actor receivers.

Afterwards, it identifies the coexisted receiver pairs in the same records (those that coexisted one time are removed). It uses such pairs to build an indirect graph where: (i) nodes correspond to receivers, and (ii) each edge relates a coexisted receiver pair. Finally, it detects the set of connected components in the obtained graph. At this level, each combination of one sender and its corresponding receiver connected component forms the actors of one sender-receivers group.
Sub-Step 2.3: Discover activity actor groups: This sup-step finds groups of actors. We define each group as a set of actors interchanging activity related emails. We suppose that an actor group is composed of a set of senders and a set of OnlyReceivers. We first generate a graph of OnlyReceivers that coexist in the same receiver list of activity emails. Then, we detect OnlyReceivers connected components. Each connected component presents then a starting point towards the construction of a group of actors. Afterwards, based on Sender-Receivers groups, we infer the set of senders interacting with each connected component and we assign them to the corresponding actor group if they coexist in the same interlocutor list. In this way, if one sender interacts with different connected components of OnlyReceivers, it will be assigned to more than one group.

7.5.2 Step 3: Discover activity actor contributions

The goal of this step (see Step 3, Figure 7.5) is to infer, for each activity, the contributions that most often each actor makes in its context. It additionally calculates, for each one, a coefficient reflecting its ratio to the overall contributions. In this way, we obtain for each activity, the contributions' distribution per actor.

Actually, actor contributions are not necessarily explicitly mentioned in emails. Additionally, the occurrence of one activity event in an email could refer to various contributions made by different actors. Let take the example of an employee that sends an email requesting performing an activity from another employee while putting his/her manager in *'CC'*. In such situation, the event of requesting activity execution implicitly refers to three actor contributions as follows: (i) request contribution made by the sender (that could be directly inferred from the activity speech act), (ii) potential contribution of execution to be made by the receiver, and (iii) observation contribution made by the manager.

For a given activity, to identify the potential contributions of its actors, this step analyses three features for each event included in its sublog: (i) speech act, (ii) actor textual indices, and (iii) future reactions (i.e. reply/forward of email event). As consequence, it mainly performs three additional sub-steps. In the first one (see Sub-step 3.1 in Figure 7.5), it determines future reactions of each event. Then, in the second sub-step (see Sub-step 3.2 in Figure 7.5), it uses these reactions combined with the event speech act and textual actor indices to infer potential contributions of event actors. Finally, it determines the overall contributions most often each actor makes in the context of the activity and it calculates the corresponding ratios. Algorithm 4 resumes these sub-steps and provides additional details on the performed operations that we explain in what follows.
Sub-step 3.1: Determine future reactions of each event (Line 8 $\rightarrow$ 12): This sub-step

Algorithm 4 Actor contribution mining from activity sublog

Input: $SL(SubLog)$, $Threads$, th_{obs} (a threshold defining if the receivers are always in observation status), $A(Actors)$
Output: C

1: $dic\_contrb\_actors = \{P : [], Ex : [], R : [], Obs : [], RI : [], I : []\}$
2: $TOs = []$, $CCs = []$, $senders = []$
3: **for** Ev **in** SL **do**
4: to=$Ev.em.to$ ▷ retrieves the receivers of the email 'em' where the event 'Ev' occurs with 'to' status
5: cc=$Ev.em.cc$ ▷ retrieves the receivers of the email 'em' where the event 'Ev' occurs with 'cc' status
6: $sender = Ev.em.sender$ ▷ retrieves sender of the email 'em' where the event 'Ev' occurs
7: $TOs = TOs + to$, $CCs = TOs + to$, $senders.append(sender)$
8: I_{values}=$ev.I_{values}$ ▷ retrieves relevant information values of the event 'ev'
9: $Threads_{IDs}$=**Filter** $(Threads, I_{values})$ ▷ select threads related to relevant information values of 'Ev'
10: $futureReacts = []$
11: **for** id **in** $Threads_{IDs}$ **do**
12: $futureReacts$=$futureReacts$+[em **for** em **in** $Threads[id]$ **if** $em.timestamp > ev.em.timestamp$] ▷ retrieves future reactions related to the event 'Ev'
13: $contrb\_act[Map[Ev.SA]].append(sender)$
14: **if** $Ev.SA == 'informationact'$ **then** ▷ Ev.SA refers to the speech act of the event 'Ev'
15: **if** $'i'$ **in** $Ev.ActorInd$ **then**
16: $contrb\_act[Ex].append(sender)$
17: **if** $'you'$ **in** $Ev.ActorsInd$ & $len(tos) == 1$ **then**
18: $dic\_contrb\_actors[Ex].append(tos[0])$
19: **if** $(Ev.SA == 'requestact')$ & $len(tos) == 1$ **then**
20: $contrb\_at[Ex].append(tos[0])$
21: $futureReacts_senders$=[$em.sender$ **for** em **in** $futureReacts$]
22: **for** r **in** tos **do**
23: **if** r *in* $futureReacts_senders$ & $Ev.SA == 'requestact'$ **then**
24: $contrb\_at[Ex].append(r)$
25: Ctr_Rec = **Counter** $(TOs \cup CCs)$, Ctr_CCs = **Counter** (CCs) ▷ Counter is a function that returns a dictionary mapping each element to the number of its occurrences in the list provided as input
26: $observers$ = [r *for* r *in* $Ctr\_CCs.keys()$ *if* $\frac{Ctr\_CCs[r]}{Ctr\_Rec[r]} > th_{obs}$ & $r \notin contrb\_act[Ex]$]
27: $contrb\_at[Obs] = [r$ **for** r **in** CCs **if** r **in** $observers]$
28: **for** $actor_i$ **in** A **do**
29: $trace\_conts$ = [[$cont$ **for** r **in** $dic\_contrb\_actors[cont]$ **if** $r == actor_i$] **for** $cont$ **in** $dic\_contrb\_actors.keys()$]
30: $trace\_conts$=$list(flatten(trace\_conts))$
31: Ctr= **Counter** $(trace\_conts)$
32: **for** $cont$ **in** ζ **do**
33: $C[actor_i][cont] = \frac{Ctr[cont]}{len(trace\_conts)}$

tracks future emails sent by actors to identify those that are related to an occurred event. These emails can be (i) of 'reply' or 'forward' relation with the event's email, (ii) of bodies containing an event relevant value, or (iii) sent to relevant event's email addresses. To this end, this sub-step retrieves first the event's relevant information values (Line 8). Then, it identifies threads containing these values or emails having forwards or reply relation with the event's email (Lines 11, 12). Finally, it retrieves emails belonging to these threads and sent at timestamps higher than thus of the event's email.

Sub-step 3.2: Determine potential actors' contributions of each event: This sub-step analyses the attributes of each event to approximate the potential contributions of its related email's interlocutors. These contributions could be:

(i) Directly deduced from event speech act (through $Map[Ev.SA]$); information, request, request information and intention acts refer to information, request, request information and planning contributions made by email sender (Line 13);

(ii) Indirectly deduced from event speech acts by exploiting the values of the other event attributes; a potential contribution of execution is assigned to: (a) an email sender if

the speech act is an information of execution and 'I' belongs to event's actor indicators (Lines 15, 16), (b) an email receiver (Lines 17 $\rightarrow$ 24) if these two conditions are satisfied;
- Cond1: The speech act is a request, or the speech act is an information and 'you' belongs to event's actor indicators;
- Cond2: The event's email is of unique receiver of sate 'To', or the receiver belongs to the list of senders of future reactions' emails (only in the case of a request speech act);

Sub-step 3.3: Determine the distribution of actors' contributions: This sub-step performs a set of aggregation operations on the overall emails' interlocutors and their potential contributions. Fist, it identifies actors of observation contribution if they are always receivers with 'CC' status without having execution contribution (lines 26 ,27). Afterwards, for each actor i, it retrieves the trace of all the occurrences of his/her contributions (Line 29). Finally, it calculates the coefficient $\lambda_{c,i}$ ($C[actor_i][cont]$) of each contribution ($cont$) using the fraction of its number of occurrences in the overall contribution trace (line 33).

7.6 Behavioral perspective discovery

The final and the fifth part in the event log mining phase focuses on discovering the behavioral perspective of each obtained BP fragment. We recall that for a given BP fragment, we describe its behavioral perspective through three model types: (i) Activity model (Definition 4.12), (ii) SA model (Definition 4.13), and (iii) Activity-SA model (Definition 4.14). Each model is composed of a set of sequencing constraints. Each constraint is of a response or co-coexistence type. It is composed of a reference event type inducing the appearance of a target event type ($\in \mathcal{A} \cup \mathcal{SA} \cup \mathcal{ASA}$). Let take the example of the following response sequencing constraint: *flow deal* $\xrightarrow{response}$ *extend deal*. The activity event type *flow deal* is the reference of such constraint. As for the event type *extend deal*, it corresponds to its target. Considering the non-controlled nature of emails and the absence of precise information referring to event timestamps, the main challenges in this part are:

- How to approximate events sequencing in emails ?
- How to minimize the effect of having more than one instance related to these events in the same email ?

In what follows, we present an overview on the behavioral perspective discovery part (Section 7.6.1). Then, we detail the main implemented steps (Section 7.6.2, Section 7.6.3).

7.6.1 Overview

We illustrate in Figure 7.6 the main steps for discovering the behavioral perspective of BP fragments. In the first one, we ***project the event log on the activities of each BP***

fragment to obtain the related sublog. In the second step, ***we generate sequencing constraint candidates*** from events appearing in emails and threads. We use two kinds of information for this purpose:

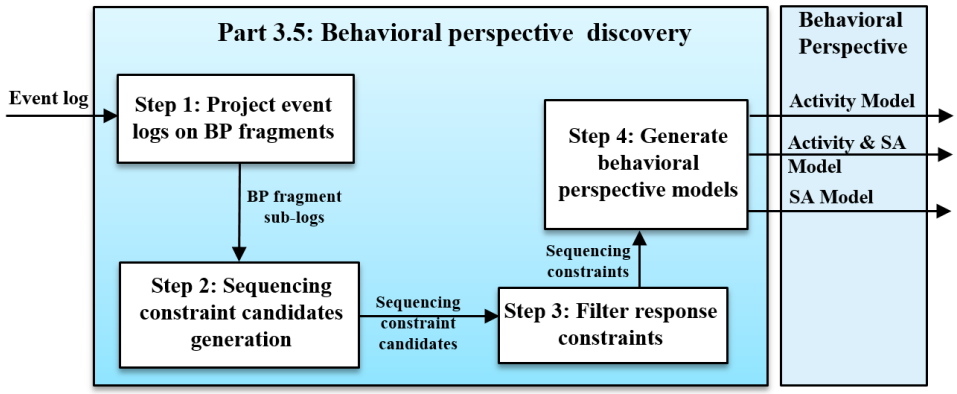


Figure 7.6: Behavioral perspective discovery: Main Steps

- Events pairs appearing successively in the same email: We consider the sequential order of events occurrences in the same email to identify pairs of successive events forming constraint candidates. We additionally consider the corresponding speech acts to define the reference and the target event in each constraint candidate.

- Events appearing in different emails but in the same threads; we consider them to deduce coexistence constraints. We also infer response constraints from events belonging to pairs of successive/sequential emails (chronologically sent).

By focusing on successive event/email pairs, this first step maximizes the possibility that: (i) Both of reference and target events of the same response constraint would be related to the same instance, and (ii) Each constraint component (i.e target / reference) induces the appearance (as a condition or as a response) of the other one.

In the second step, based on the constraint frequency notion, we ***filter the obtained constraint candidates*** to keep those of high confidence and remove non-significant ones (e.g. those of low frequency).

Finally, in the third step, we ***generate the behavioral perspective models*** from the filtered sequencing constraints by organizing them into a graphical representation.

It is important to note that a coexistence constraint between a reference and a target event is equivalent to a causality relation (Definition 7.3) between the two events. That is why we discover them with the same manner as it was detailed in Section 7.3. In such case, the second step of filtering constraint candidates is ensured through the conditional probability of the corresponding causality relation. For activity coexistence constraints, they are directly deduced from activity-activity causality relations that have induced the appearance of the

related BP fragment. In what follows, we will focus on the second and the third steps for discovering response sequencing constraints.

7.6.2 Generate response sequencing constraint candidates in Step 2

In this step, we generate three response sequencing constraint candidates types in accordance with the behavioral models that we aim to discover. These types are: (i) Activity constraint candidates, (ii) Activity-SA constraint candidates, and (iii) SA constraint candidates. The discovery of each constraint candidate type is detailed in what follows.

7.6.2.1 Activity sequencing constraints for activity model discovery

For discovering activity sequencing constraints of response type, it is required to identify the relative chronological order of activity events. The main challenge here is how to approximate this order in the same email (in the case of multiple events appearing in the same email) and in the same thread (in case where they belong to different emails). In fact, emails do not only present execution traces of activities, they could also introduce activities that are not executed, under execution or to be executed. Therefore, timestamps of activity events do not generally match the timestamps of emails where they occurred. Hence, estimating the actual timestamps of activity events appearing in emails to be able to order them in ascending chronological order in each thread would not be always feasible.

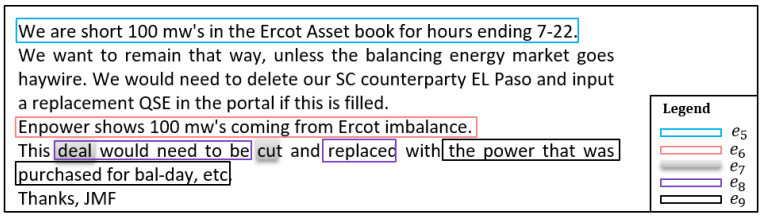
Given this challenge, we propose to approximate the relative sequencing order of activity events from the textual contents of emails and their sequencing in threads (rather than requiring their exact timestamps). To this end, we perform these two iterations:
- ***Iter1: Mine response sequencing constraint candidates from emails containing multiple activities:*** These constraints actually reflect how a BP fragment is described by employees themselves in emails. Employees could use emails as support to explain/request to/from other employees actions to be done as a response to some events. The goal is to exploit such emails in order to mine employees business expertise in the context of one BP fragment. Algorithm 5 describes how to mine response sequencing constraint candidates from emails containing multiple activities. First, it regroups events by email ID (line 2). Then, for each email (having more than one occurring activity), it sorts its events, in increasing order, according to the position of appearance of their related occurring activities (line 6). Second, from each pair of successive events appearing in the same email, a sequencing constraint candidate is formed. Speech acts of events are considered to ensure that if their activities occurrences were actually performed in different tenses (which means one is in the past tense in case of *'information'* speech act and the other is in the future tense in the case of *'request'*, *'request information'* or *'intention'* speech act), the event of past tense will be privileged to be the reference event and thus of future event to be the target event. In fact, activities performed in the past come logically before activities that will be performed in the future. This step retrieves then the related speech acts and activity names (lines 7 $\rightarrow$ 10). Afterwards, if the speech act of the first event is not an information about an executed activity

Algorithm 5 Activity Constraint Candidates Generation From Emails

Input: $FPsublog$
Output: $allConstraintsCandidates$
1: $allConstraintsCandidates = []$
2: $eventsPerEmail$=$GroupEventPerEmail$ $(FPsublog)$ ▷ GroupEventPerEmail returns a dictionary that organizes events per email ID, for each email ID, a list of events is associated
3: **for** $emailID$ **in** $eventsPerEmail.keys()$ **do**
4: $emailevents = eventsPerEmail[emailID]$
5: **if** $len(emailevents) > 1$ **then**
6: $sortedEvents = SortByPositions(emailevents)$ ▷ SortByPositions sorts a list of events into asscending order according to the position of occurrences of their related activities in the email
7: $SuccessivePairs$=$[(sortedEvents[i], sortedEvents[i+1])$ **for** i **in** $[0, len(sortedEvents) - 1]$
8: **for** $(event_1, event_2)$ **in** $SuccessivePairs$ **do**
9: $SA_1 = event_1[2]$, $SA_2 = event_2[2]$
10: $activity_1 = event_1[1][0]$, $activity_2 = event_2[1][0]$
11: **if** $SA_1 \neq$ *'information'***AND** $SA_2 ==$*'information'* **then**
12: $constraint = activity2 +$ *'BEFORE'*$+activity1$
13: **else**
14: $constraint = activity_1 +$ *'BEFORE'*$+activity_2$
15: $allConstraintsCandidates.append(constraint)$

(i.e. the activity will be performed in the future) and the speech act of the second event is an information about execution (i.e. the activity was executed in the past), the event order will be switched when forming the related constraint candidate (lines 11&12). Otherwise, the order of events according to their position of appearance will be kept (lines 13&14).

Let take the example of the email in Figure 1.3 of which we show an extract concerning its main body in Figure 7.7 (a). Let consider the events e_5, e_6, e_7, e_8 and e_9 appearing in it (see Figure 7.7 (b) extracted from Figure 3.5). We frame/highlight the expressions where these events occur as explained in the legend of Figure 7.7 (a). If we consider that only these



(a) Email main body

Event	Act_o			SA	At_{ind}	I_{values}	em					Th_{id}
Ev_ID	AN_{occ}	BD_{occ}	BC_{occ}				ID	times tamp	Sender	Recipients	Con vIDs	
e_5	('open short position', ['short', '100', 'mws'], [1, 2, 3])	{('numeric mw',['numeric','100'], [2,3]), ('hour0end numeric numeric',['hour ending','7', '22'], [5,6]), ('orgname', ['Ercot Asset'], [4]), ('orgname', ['El Paso'],[20]), ('orgname', ['Empower'], [26]), ('numeric mw',['numeric','100'], [28,29]), ('orgname', ['Ercot'], [30]), ('orgname', ['bal-day'], [38])}	{}	'information'	['we']	[]	'<23535150.1075850236920.JavaMail.evans@thyme>'	'Wed, 26 Sep 2001 10:19:15 -0700 (PDT)'	'm..forney@enron.com'	['joe.capasso@enron.com', 'l..day@enron.com', 'joe.errigo@enron.com', 'alexander.mcelreath@enron.com', 'jeffrey.miller@enron.com', 'steve.olinde@enron.com', 'eric.saibi@enron.com']	'C3'	['Th_3']
e_6	('display deal', ['shows', '100', 'mws'], [27, 28, 29])			'information'	['enpower']							
e_7	('cut deal', ['cut', 'deal'], [33, 31])			'intention'	[]							
e_8	('replace deal', ['replace', 'deal'], [34, 31])			'intention'	[]							
e_9	('purchase power', ['power', 'purchased'], [36, 35])			'information'	[]							

(b) Event log extract related to the email main body in (a)

Figure 7.7: Illustrative example for response sequencing constraints generation from an email main body

events appear in the email main boy of Figure 7.7 (a), we generate the following sequencing constraint candidates:
- *'open short position* $\xrightarrow{response}$ *display deal'*: the order of events is privileged here because *'short power'* appeared before *'display deal'* and their related speech acts (i.e. information) refer both to same tense (i.e. past) ;
- *'display deal* $\xrightarrow{response}$ *cut deal'*: the order of events is privileged here because *'display deal'* with past tense appeared before *'cut deal'* with future tense (which is logically true);
- *'cut deal* $\xrightarrow{response}$ *replace deal'*: the order of events is privileged here because *'cut deal'* appeared before *'replace deal'* and their related speech acts (i.e. intention) refer both to the same tense (i.e. future);
- *'purchase power* $\xrightarrow{response}$ *replace deal'*: the tense is privileged here as *'replace deal'* with future tense appeared before *'purchase power'* with the past tense;
- ***Iter2: Mine response sequencing constraint candidates from activities of the same threads:*** If an email informs about the occurrence of one event, these constraints report what activities could be mentioned in next emails to be (or that was) performed as a response. To generate them, each pair of successive emails in each thread is analysed in the way that an activity with *'information'* speech act in the first email will form a constraint with others activities of the next email. The goal here is to find out what activities to be performed following the information about the execution of one activity in one email. That's why only events of *'information'* speech act are considered in the first email of each successive email pair. Algorithm 6 describes how to mine sequencing constraint candidates of response type from activities of the same threads. It organizes events by threads IDs and subsequently by email IDs for each thread (line 2 → 5). Then, it generates the set of successive emails pairs for each thread (line 7). After that, for each pair, it only considers events of *'information'* speech act in the first email. Each activity of these events will form constraint candidates with activity events of next emails (lines 11 → 15).

Algorithm 6 Activity Constraint Candidates Generation From Threads

Input: $FPsublog$
Output: $allConstraintsCandidates$
1: $allConstraintsCandidates = []$, $UniqueEmailsPairs = []$
2: $eventsPerThread$=$GroupByThread$ $(FPsublog)$ ▷ GroupByThread returns a dictionary that organizes events per thread ID
3: **for** $threadID$ **in** $GroupByThread.keys()$ **do**
4: $threadEvents = eventsPerThread[threadID]$
5: $eventsPerEmail = GroupEventPerEmail(threadEvents)$, $emailsIDs = eventsPerEmail.keys()$ ▷ GroupEventPerEmail returns a dictionary that organizes events per email ID
6: **if** $len(emailsIDs) > 1$ **then**
7: $SuccessiveEmailsPairs = []$
8: **for** i **in** $[0, len(emailsIDs) - 1]$ **do**
9: **if** $(emailsIDs[i], emailsIDs[i+1]) \notin UniqueEmailsPairs]$ **then**
10: $SuccessiveEmailsPairs.append(emailsIDs[i], emailsIDs[i+1])$
11: $UniqueEmailsPairs.append(emailsIDs[i], emailsIDs[i+1])$
12: **for** $(emailID_1, emailID_2)$ **in** $SuccessiveEmailsPairs$ **do**
13: **for** $event_1$ **in** $eventsPerEmail[emailID_1]$ **do**
14: $SA_1 = event_1[2]$
15: **if** $SA_1 ==$ *'information'* **then**
16: **for** $event_2$ **in** $eventsPerEmail[emailID_2]$ **do**
17: $activity_1 = event_1[1][0]$, $activity_2 = event_2[1][0]$
18: $constraint = activity1$+*'BEFORE'*+$activity2$
19: $allConstraintsCandidates.append(constraint)$

Taking the example of emails $email_1$, $email_2$, $email_3$ and $email_4$ belonging to the same thread as illustrated in Figure 7.8, we obtain the following set of successive email pairs: ($email_1$, $email2$), ($email_2$, $email_3$) and ($email_3$, $email_4$). These email pairs generate the following sequencing constraint candidates:
- *'flow deal* $\xrightarrow{response}$ *roll deal'* because *'flow deal'* occurred in $email_1$ with *'information'* speech act and *'roll deal'* occurred in its next email (i.e. $email_2$) ;
- *'roll deal* $\xrightarrow{response}$ *flow deal'*, *'roll deal* $\xrightarrow{response}$ *extend deal'*, and *'roll deal* $\xrightarrow{response}$ *create deal'* because *'roll deal'* occurred in $email_2$ with *'information'* speech act and *'flow deal'*, *'extend deal'* and *'create deal'* occurred in the next email (i.e. $email_3$) ;
- *'flow deal* $\xrightarrow{response}$ *extend deal'* because *'flow deal'* occurred in $Email_3$ with *'information'* speech act and *'extend deal'* occurred in its next email (i.e. $email_4$);

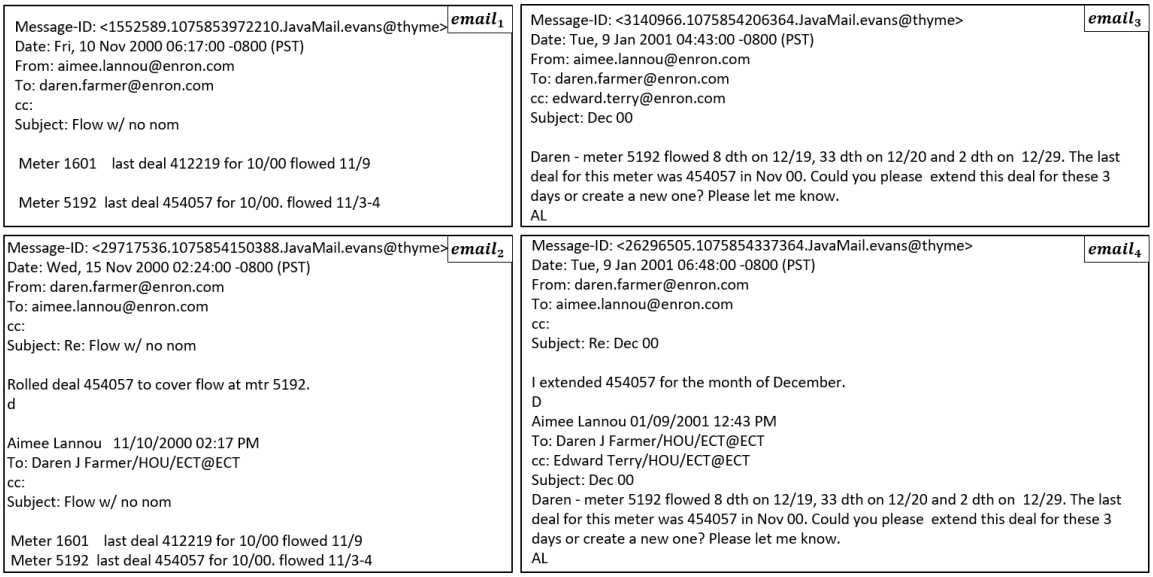


Figure 7.8: Example of Enron emails belonging to the same thread

7.6.2.2 Activity & SA response sequencing constraint candidates for activity & SA model discovery

Activity & SA response sequencing constraints reflects how emails support the execution of BP fragment activities. The identification of such constraints includes the recognition of: (i) the actual event labels that describe the occurrences of activities in one email when sending it, and (2) their related sequencing order. For event labels, they correspond to activity names of the occurred activities concatenated with their speech acts (i.e. specified activities). As for approximating their related response sequencing constraints, we consider that for each event, related to the occurrence of one activity specified by its speech act, it logically has the same timestamp as the email where it has appeared (e.g. the event of requesting deal extension has the same timestamp of the email containing extend deal occurrence with a $'request'$ speech act). This means that by focusing on two successive emails appearing in the same thread,

events of the first email will have a timestamp lower than events appearing in next email. As described in Algorithm 7, this sub-step first generates pairs of successive emails from threads (line 2 → 7). Then, constraint candidates are generated in the way that for each pair and for each couple of events belonging to its emails (one in each email), a constraint is formed while its target event belongs to the first email and its reference event belongs to the second email (line 8 → 13).

Algorithm 7 Activity-SA Constraint Candidates Generation From Threads

FPActivities,
Input: *FPsublog*
Output: *allConstraintsCandidates*
1: *allConstraintsCandidates* = []
2: *eventsPerThread*,*ThreadsIDs*=*GroupByThread* (*FPsublog*)
3: **for** *threadID* **in** *ThreadsIDs* **do**
4: *threadEvents* = *eventsPerThread*[*threadID*]
5: *eventsPerEmail*, *emailsIDs* = *GroupEventPerEmail*(*threadEvents*)
6: **if** *len*(*emailsIDs*) > 1 **then**
7: *SuccessiveEmailsPairs*=[(*emailsIDs*[*i*], *emailsIDs*[*i* + 1]) **for** *i* **in** [0, *len*(*emailsIDs*) − 1]
8: **for** (*emailID*$_1$, *emailID*$_2$) **in** *SuccessiveEmailsPairs* **do**
9: **for** *event*$_1$ **in** *eventsPerEmail*[*emailID*$_1$] **do**
10: **for** *event*$_2$ **in** *eventsPerEmail*[*emailID*$_2$] **do**
11: $SA_1 = event_1[2]$, $SA_2 = event_2[2]$
12: $activity_1 = event_1[1][0]$, $activity_2 = event_2[1][0]$
13: *constraint* = *activity*1+'___'+SA_1+'BEFORE'+*activity*2+'___'+SA_2
14: *allConstraintsCandidates.append*(*constraint*)

For the SA response sequencing constraint candidates used for SA model discovery, they depend on a given activity name to mine the related sequencing of email uses. Consequently, they are deduced from the Activity-SA sequencing constraint candidates by projecting them on a given activity name.

7.6.3 Filter response sequencing constraint candidates in Step 3

By focusing on successive events pairs or emails pairs to form sequencing constraint candidates (Step 1, Figure 7.6), we aim to maximize the possibility that both of reference and target events of the same constraint would be related to the same instance. However, such assumption would not always be verified as well as other assumptions adopted to handle events of the same speech act (e.g. activity events of the same speech act and of the same email will be chronologically sorted according to their appearance order in the email). This would generate contradictory constraint candidates (i.e. two constraints with switched reference and target event type) or non-significant constraints of low number of occurrence. Taking the example of the sequencing constraint candidates generated from the successive email pairs ($email_1$, $email_2$) and ($email_2$, $email_3$) (see Figure 7.8), *'flow deal'* $\xrightarrow{response}$ *'roll deal'* and *'roll deal'* $\xrightarrow{response}$ *'flow deal'* are two contradictory constraints.

At the level of this third step (i.e. Step 3, Figure 7.6), we filter the obtained candidate constraints to keep significant ones by only preserving those:(i) that have occurred more than a minimum number of occurrence $th_{N_{constOcc}}$ (a user parameter) to avoid obtaining constraints that have rarely appeared, and (ii) that have a high confidence measure (determined accord-

ing a user parameter $th_{Confd_{const}} \in [0, 1]$).
To formally define the expression of such confidence measure, let consider the following notations:

- $Const_{ij}$ be a response sequencing constraint of reference i and target j;
- $Const_{ji}$ denote its contradictory constraint of reference j and target i;
- $card(Const_{ij})$ is the number of occurrence of the constraint candidate $Const_{ij}$ accross emails or threads;

The confidence measure $Confd(i, j)$ of the response sequencing constraint $Const_{ij}$ is defined according to the formula 7.2:

$$Confd(i, j) = \frac{card(Const_{ij})}{card(Const_{ji}) + card(Const_{ij})} \tag{7.2}$$

The confidence measure of a response sequencing constraint has a real value that varies between 0 and 1. The closer the value is to 1, the fewer times its contradictory constraint appears through threads or emails. This means that whenever the corresponding event types appear, they are likely to induce the same response sequencing constraint candidate. This would refer to the high relevance degree of the corresponding constraint.

7.7 Experiment results

This section presents the results of the experiments that we have carried out to study the performances of our event log mining phase. In these experiments, we were based on the same email dataset (extracted from the Enron dataset) that we have adopted in the previous chapter (see Table 5.2). More precisely, we started from the generated event log that we described in Section 6.5.3 and whose we give an overview in Table 6.4. We recall that this event log is composed of 3120 events that report the occurrences of 102 activities belonging to 1287 threads. The performed experiments in this section cover key parts related to the event log mining phase; (i) functional and data perspectives' discovery including artifacts and BP fragments (Section 7.7.1), (ii) organizational perspective discovery (Section 7.7.2), and (iii) behavioral perspective discovery (Section 7.7.3). For each part, we highlight: (i) our methodology for annotating the evaluation data, (ii) the experiments and the results that we carried out, and (iii) our visualization for the obtained results.

7.7.1 Functional and data perspectives discovery

This part evaluates our steps for discovering the functional and the data perspectives including artifacts and BP fragments. To this end, it first presents the related BP elements that we

have manually identified and the methodology that we have adopted during this manual annotation process. Then, it details our experiment results related to: artifact discovery and BP fragment discovery. Finally, it highlights the data perspective visualization that we introduced and it discusses some results.

7.7.1.1 Annotation Methodology

To manually define BP elements related to the functional and data perspectives from our evaluation dataset, we proceeded as follows:
- **For manually defining artifacts, their artifact-artifact and artifact-activity relations:** Various level of resolution/granularity could be actually adopted for this purpose. This results in different overlapping re-partitions of activities that would vary from one annotation to another. Given the example of the trading operations applied on gas energy, a meter is allocated to each gas deal to capture gas flow ratio while receiving/delivering gas. Two sets of artifacts of different resolutions could be defined to describe data manipulated by these operations:
i) A set S_1 = {Gas deal artifact} of low resolution while the gas deal attributes include a meter identifier, a gas volume and a deal identifier;
ii) A set S_2= {Gas deal artifact, Meter artifact} of higher resolution while the meter artifact is characterized by an identifier and a volume attributes and has an association relation with gas deal artifact characterized by its identifier;
In such example, activity allocation to artifacts will vary from S_1 to S_2; activities acting on meter and gas deal will have one association with one artifact in case of S_1 and two associations with two artifacts in case of S_2. This means that the manually defined overlapping re-partitions of activities will vary from S_1 to S_2.

To identify artifacts appearing in our evaluation dataset and their lowest granularity degree, we adopted the following strategy composed of the following sequence of operations:

- OP1: Define the manipulated business data by the activities present in our event log;
- OP2: Identify each set of common business data manipulated by at least two activities;
- OP3: Associate an artifact to each set of business data found in common between a number of activities;
- OP4: For artifacts that are composed of one business data, check if they could be kept (e.g. meter artifact composed of meter identifier business data) or must be logically merged with another artifact (e.g. an artifact composed of price business data);

Figure 7.9 shows the different artifacts that we have manually identified. Each artifact is represented by an informational entity composed of a name (e.g. power deal) and the set of associated business data referring to its attributes (e.g. power quantity, hour ending and zone). These informational entities are related with a set of associations referring to artifact-artifacts relations. The manual designation of activities to artifacts (i.e. artifact-activity

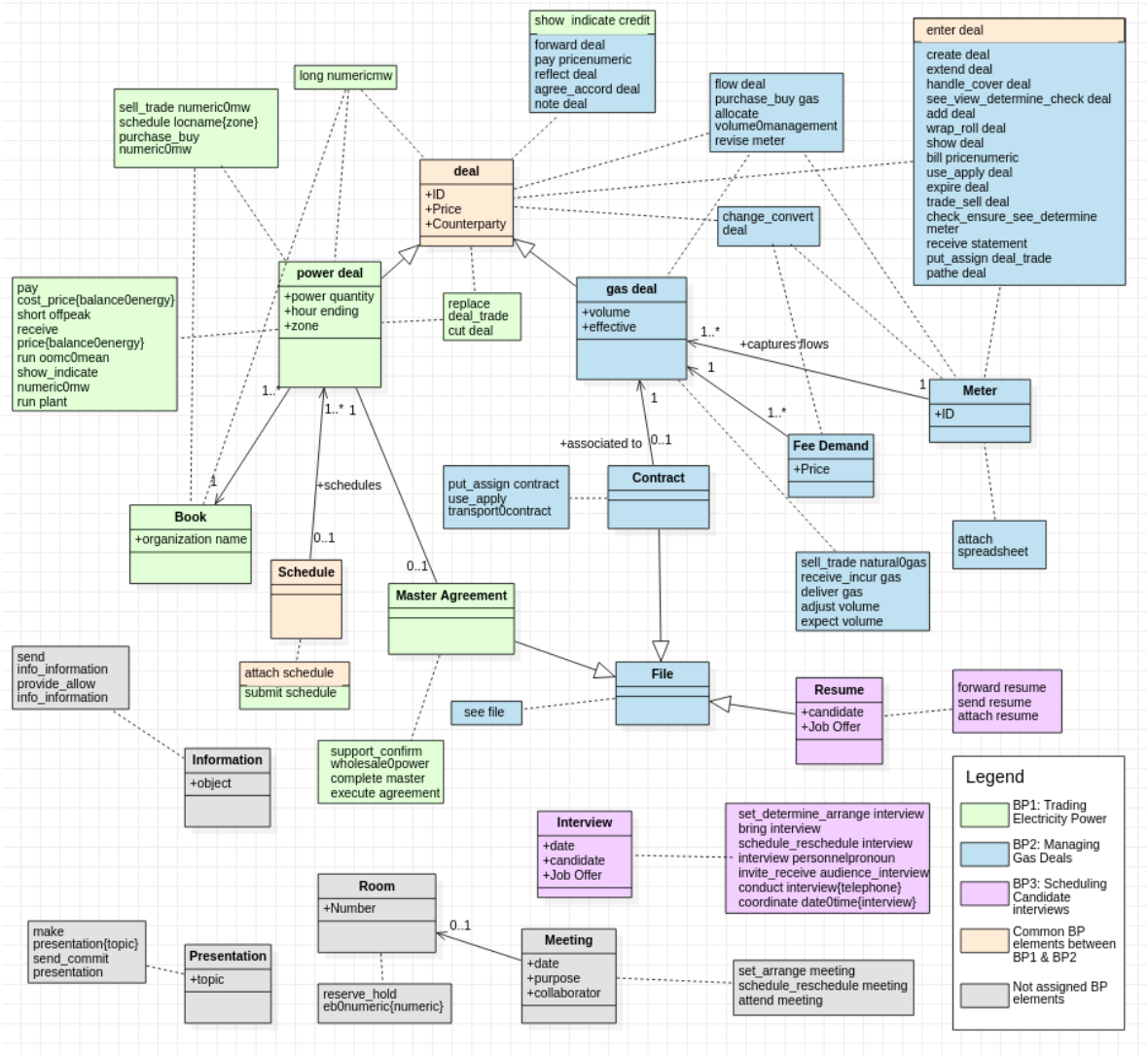


Figure 7.9: Manually Identified data and functional perspectives

relations) are shown in the same figure through the dashed lines. These lines relates each set of activities to the artifacts that they manipulate, for example, the set of activities {replace deal_trade, cut deal} were manually designated to the artifacts 'deal' and 'power deal'.

- **For manually defining BP fragments:** We relied on our own understanding of the emails and the underlying BP. We identified dependent activities executed by the same group of actors/communicators to find BP fragment types and their related elements. Three BP fragments types were then defined: (i) BP_1: Trading electricity power, (ii) BP_2: Managing gas deals, and (iii) BP_3: Scheduling candidate interviews. We have also defined a category of BP elements that do not appear to belong to any BP fragments. Figure 7.9 illustrates, as detailed below, this manual designation of BP elements to the identified categories and BP fragments:

- Shapes colored in green refer to BP elements belonging to BP_1;
- Shapes colored in blue refer to BP elements belonging to BP_2;

- Shapes colored in purple refer to BP elements belonging to BP_3;
- Shapes colored in orange refer to BP elements belonging to BP_1 and BP_2;
- Shapes colored in grey refer to BP elements that were not designated to any BP fragment (e.g. the 'Meeting' and 'Information' artifact and their related activities).

7.7.1.2 Artifact discovery: Experiments and Results

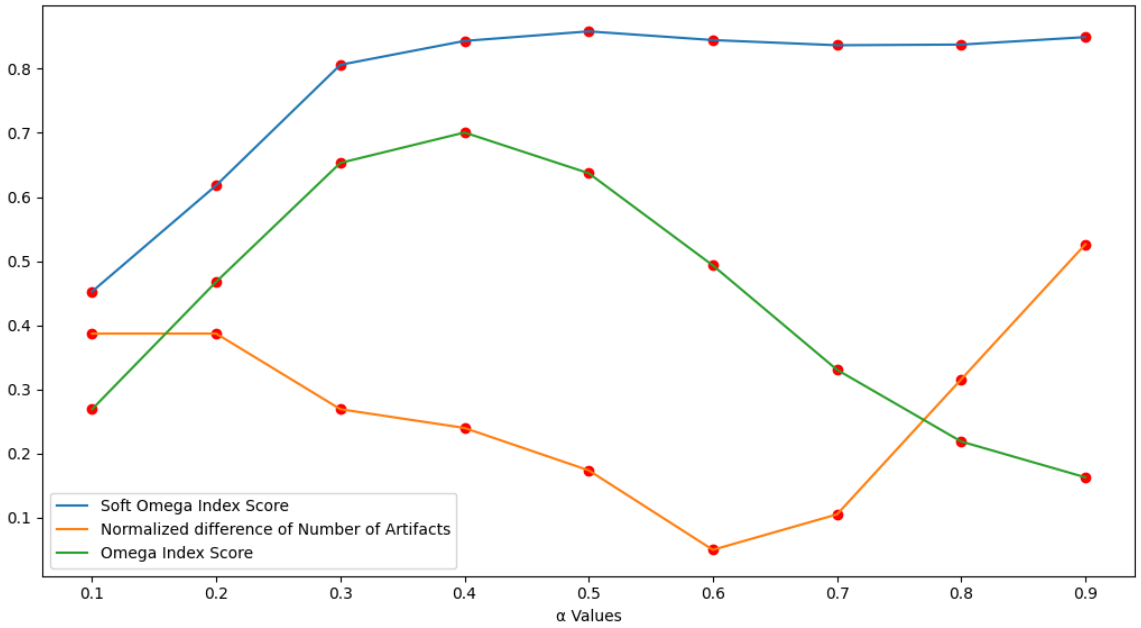
To evaluate our artifact discovery step, we have mainly employed three metric types:
(1) Normalized artifact number difference: This measure aims to calculate the difference between the discovered and the annotated number of artifacts This error difference (E) has the following expression:

$$E = \frac{abs(N_d - N_a)}{max(N_d, N_a)} \tag{7.3}$$

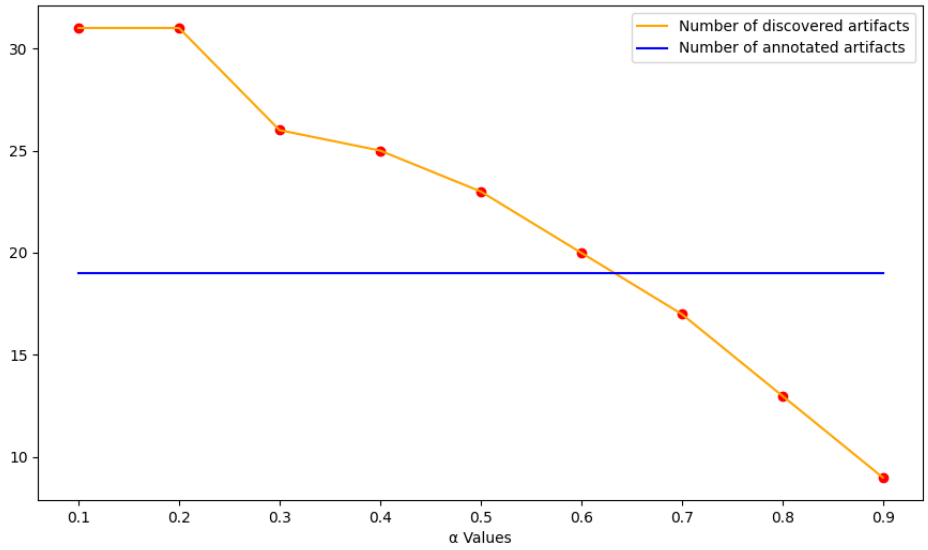
where N_d and N_a denote the number of discovered and annotated artifacts respectively. We aim from this metric to approximate the difference between the manually identified resolution degree of artifacts and the discovered one.
(2) Omega index: [19] Omega index was introduced to evaluate overlapping clustering. It is the equivalent of ARI (Adjusted Random Index) used for evaluating non-overlapping clustering. It counts the number of pairs of elements present in exactly the same number of clusters as in the number of manually annotated categories (zero number is also considered). However, it does not consider pairs present in different number of clusters and categories.
(3) Soft Omega Index: [59]: It is an extension of Omega Index. It additionally takes into account pairs present in different number of clusters and categories. This is ensured by normalizing the number of occurrences of each pair in all clusters of one clustering (manually defined or automatically discovered) by the larger number of occurrences in the other clustering. Soft Omega Index is a measure that varies between 0 and 1.

The closer the measure value of Omega index/ Soft Omega index is to 1, the more similar the manually annotated categories and the discovered clustering are. We use Omega index and Soft Omega Index in this evaluation step because they are applicable even in the absence of perfect matching between annotated categories and the discovered clusters (in terms of number also). This is useful in our case since artifacts could be defined or discovered according to various level of granularity/resolution as previously explained. Taking the example of the two sets of the artifacts S_1 and S_2, the related activity pairs will be present in at least one artifact cluster in both annotated and discovered partitions of clusters. This induces a similarity degree between the two partitions even with different resolutions and number of clusters.

Our artifact discovery step mainly depends on a predefined user threshold $\alpha \in]0, 1]$ that defines the minimum degree of pairwise similarity between activities of the same artifact activity group (Definition 7.2). Figure 7.10(a) shows the evolution of Omega Index (see the



(a) Clustering metrics evolution per α values



(b) Number Of Artifacts Evolution per α value

Figure 7.10: Overlapping Clustering Metric values evolution per α value

blue curve) and Soft Omega Index (see the green curve) scores while varying α in [0,1] (with 0.1 step). It shows also the evolution of the normalized difference of the number of annotated and discovered artifacts according to the same threshold variation (see the orange curve). The overall curves show maximum performance in the vicinity of 0.4 value of α. At this latter, Omega Index score is maximal (is equal to 0.7), the soft Omega Index is equal to 0.84 and the normalized difference is equal to 0.25. Figure 7.10(b) shows the evolution of the number of discovered artifacts per the same α values and how this number decreases while α value increases. This reflects that at high (low) α values, the resolution of the discovered artifacts decreases (increases) and the probability of missing the discovery of an existing

artifact increases (decreases). This leads to the increase of the normalized difference value between the discovered and the annotated number of artifacts at low and high α values as shown is Figure 7.10 (a).

7.7.1.3 BP fragments discovery: Experiments and Results

To compare the obtained BP fragments with the annotated ones, we adopted the following formula for calculating F1 scores [88] (adapted to the context of an overlapping clustering);

$$F1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2card(Pr_i \cap Dv_i)}{card(Pr_i) + card(Dv_i)} \tag{7.4}$$

where n denotes the number of unique activities and Pr_i and Dv_i denote the set of BP fragments predicted and discovered for an activity i.

Our BP fragment discovery step mainly depends on two main confidence measures: (1) γ defining from which value an association rule is considered a causality relation, and (2) β defining from which value an activity could be added to an initial group. We vary these two parameters (with step of 0.1) and we show the evolution of **F1 scores** in Figure 7.11. As for the number of activity pairs required for merging activity groups while ensuring that they share the same actors and the same business context, we set it to the minimal value 1 (which logically maximize the sensitivity of our results to erroneous group merging).

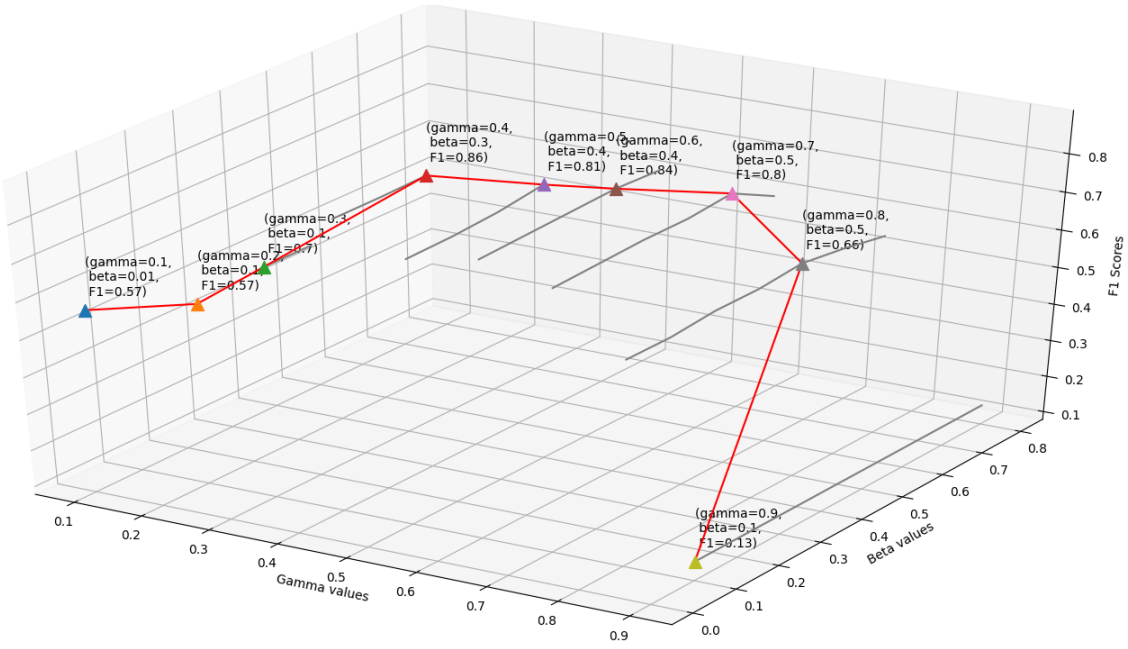


Figure 7.11: F1 scores evolution per γ and β values

Each gray curve in Figure 7.11 corresponds to a γ value. It shows the evolution F1 score when varying the value of $\beta < \gamma$. The maximum F1 score value for each curve is marked by a

triangular and colored point (e.g. F1=0.71 for $\gamma = 0.3$ and $\beta = 0.1$. The overall maximum F1 score values are related by a red curve. This curve shows that F1 score reaches its maximum value for a gamma value equal to 0.4, than, it decreases and tends to zero for high gamma values. This evolution can be explained by the following reasons:
1) For low γ values, association rules of low confidence measure are considered to form causality relations. This induces the possibility of merging different BP fragments activities into one connected component. Figure 7.12 shows an example of connected components obtained for $\gamma = 0.2$. Each circular node in the figure refers to an activity and each edge connects two activities having a causality relation sharing at least one artifact and one actor. The figure

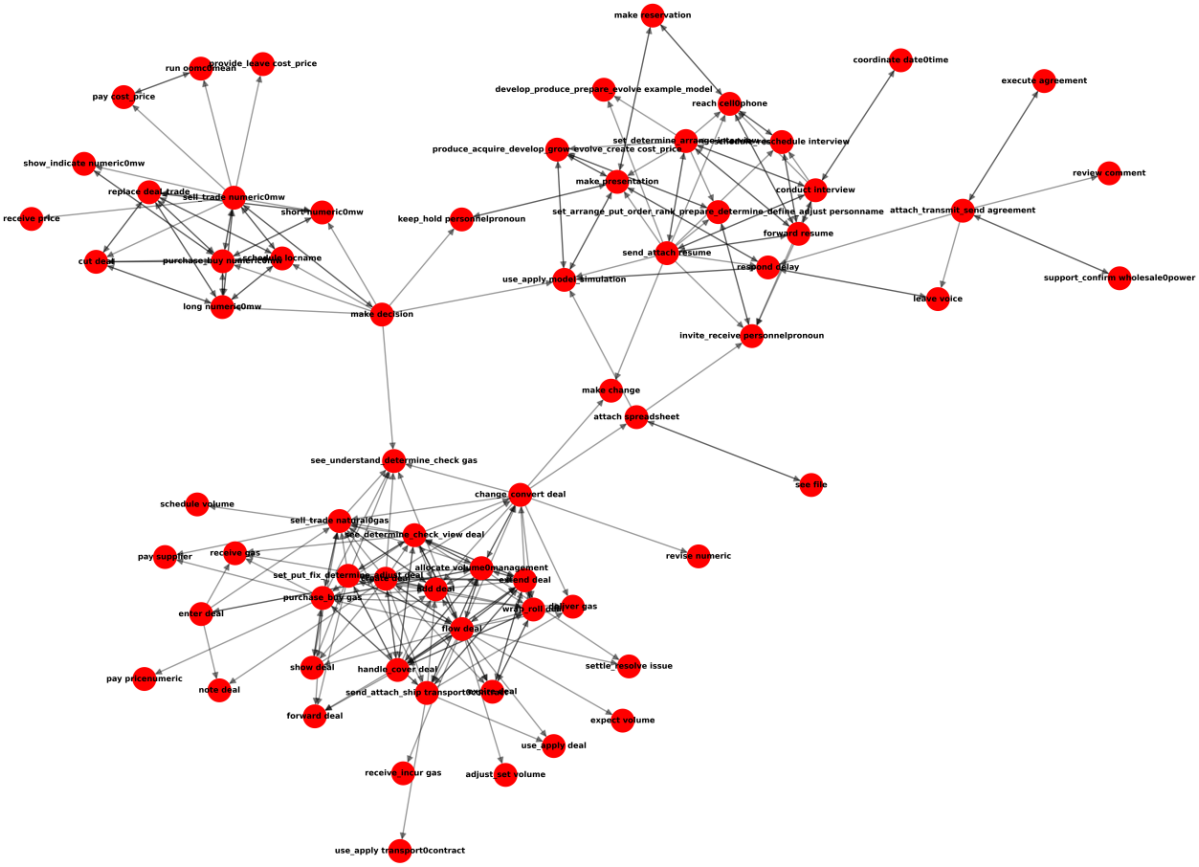


Figure 7.12: Connected Components for $\gamma = 0.2$

shows how activities of two different BP fragments (i.e. trading electricity power & managing gas deal) are grouped into the same connected group (colored in red);
2) For high γ values, association rules form causality relations only if they are of high confidence measure. This induces the disappearance of some BP fragments. Figure 7.13 shows an example of connected components obtained for $\gamma = 0.9$ where only one group referring to organizing interviews is kept through one causality relation (formed by *'conduct interview'* and *'coordinate date0time'*);
3) For γ values in the neighborhood of 0.5, association rules of intermediary confidence measures are considered to form causality relations, this induces better recall and less accuracy error while detecting BP fragments. Figure 7.14 shows the obtained connected components

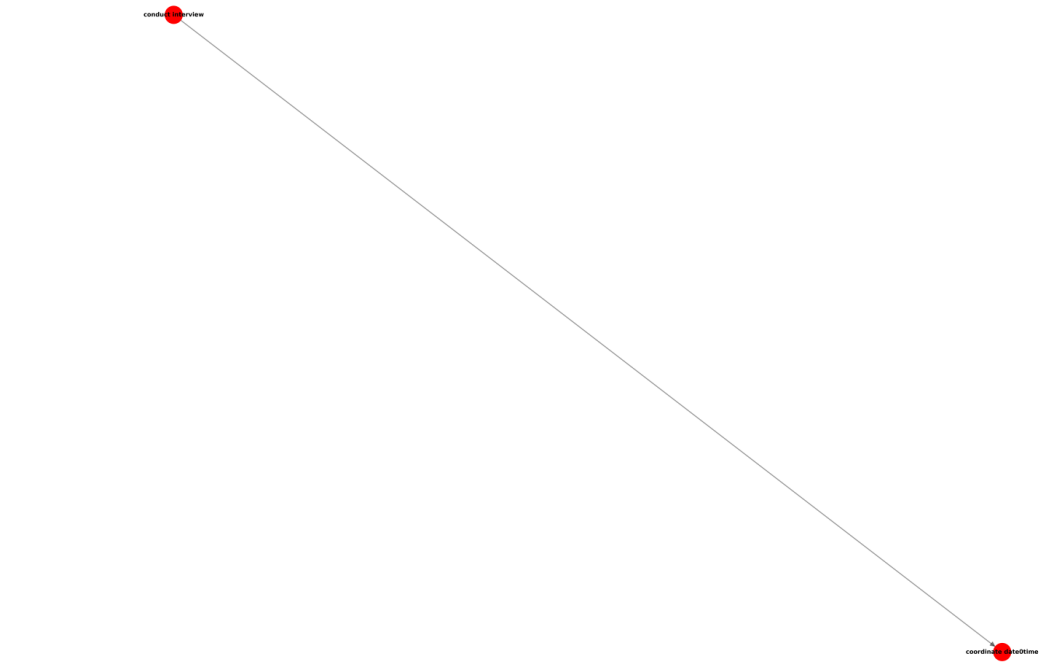


Figure 7.13: Connected Components for $\gamma = 0.9$

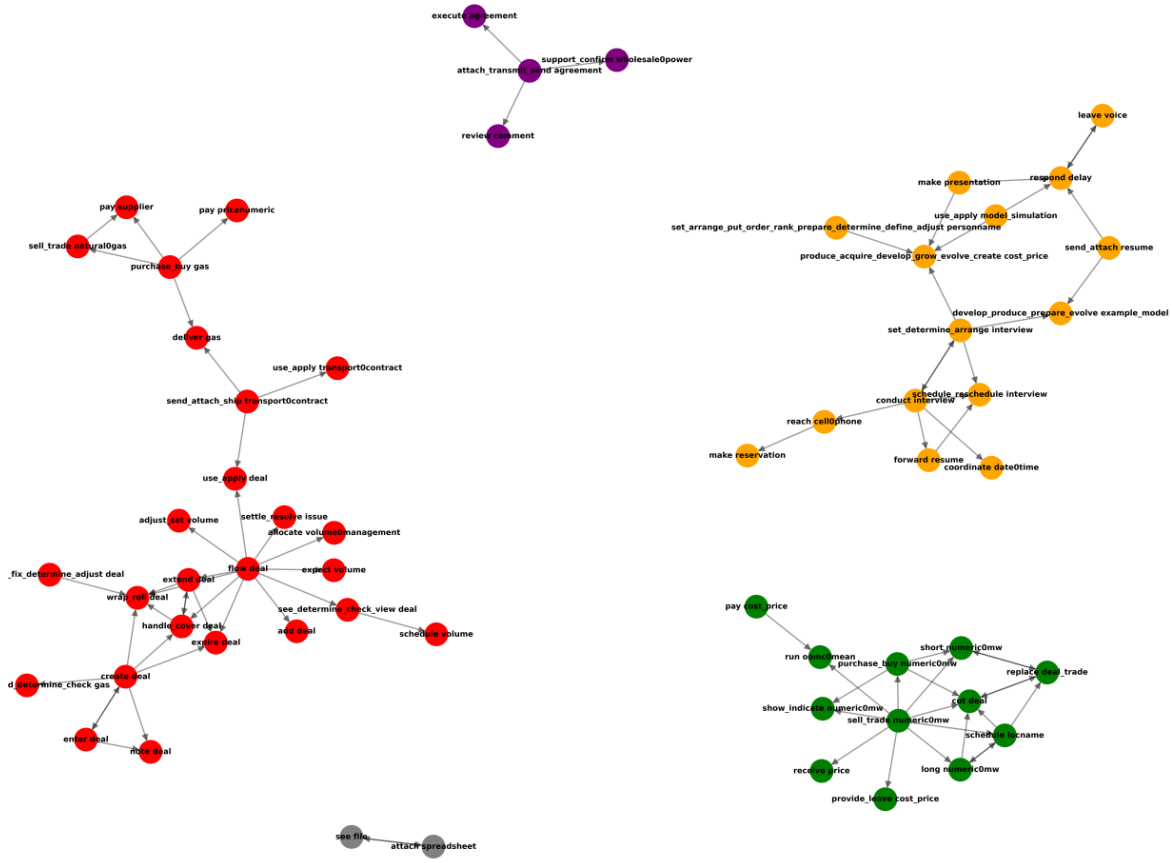


Figure 7.14: Connected Components for $\gamma = 0.4$

for $\gamma = 0.4$ where three of them refer to our three annotated BP fragments; activity nodes of trading electricity power are colored in green, those of managing gas deals are colored in red and those of organizing interviews are colored in orange;

7.7.1.4 Data perspective Visualization: Visualization and discussion

We conceived a visualization tool that generates a graph resuming the data perspective of each BP fragment. Figure 7.16 and Figure 7.15 present examples of graphs that could be obtained for BP_1 and BP_2 where $\alpha = \gamma = 0.4$. Each graph is composed of the following elements:

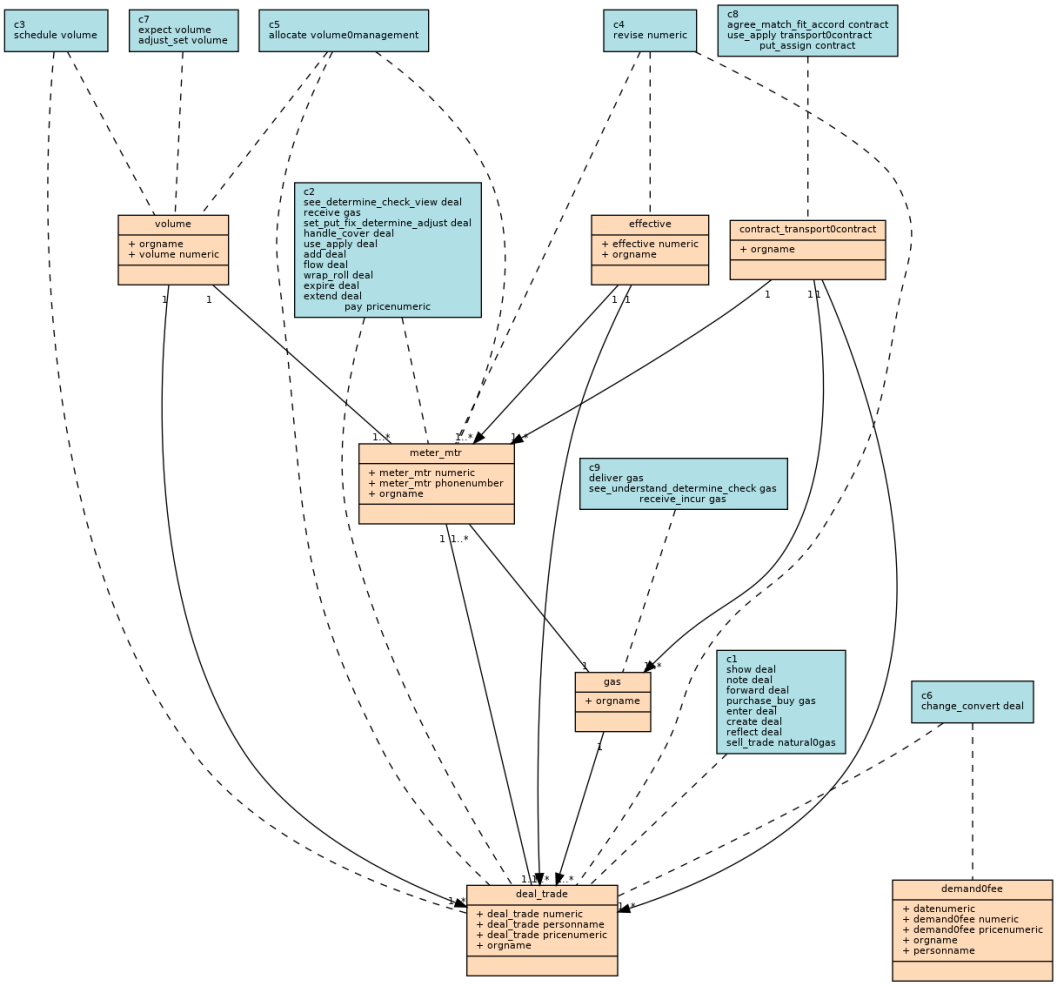


Figure 7.15: Data perspective of Managing Gas deals BP fragment

- Artifact rectangular nodes colored in orange. Each node is composed of two lines; in the first line artifact name (e.g. meter_mtr in Figure 7.15) is printed and in the second line artifact attributes (e.g. meter_mtr numeric, orgname) are enumerated;

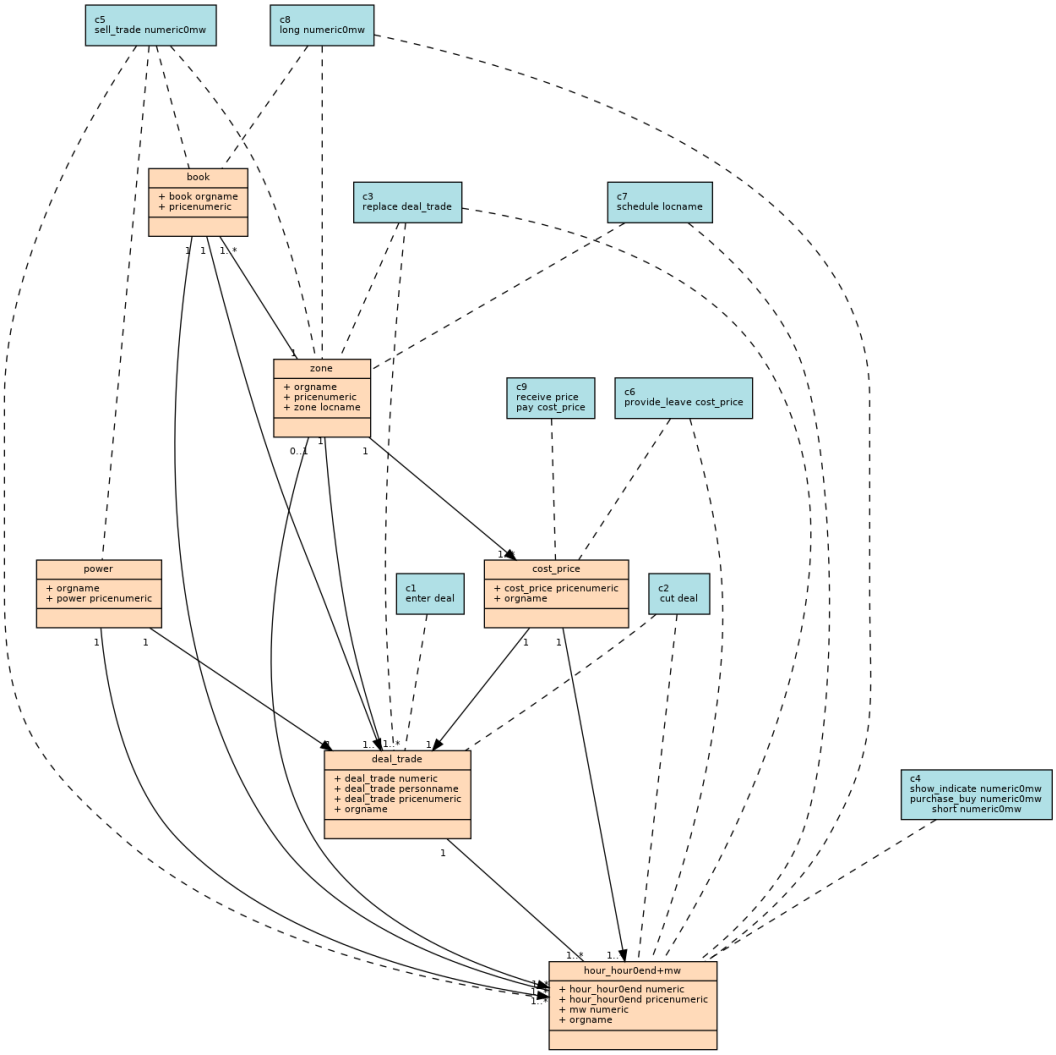


Figure 7.16: Data perspective of Purchasing Electricity Power BP fragment

- Activity rectangular nodes colored in blue: each node regroups a set of activities acting on the same set of artifacts;
- Dashed edges linking each activity group to the set of artifacts nodes that acts on them, e.g. the node c4 in Figure 7.16 composed of one activity replace deal_trade acts on three artifacts: zone, hour_hour0end+mw and deal_trade;
- Directed edges linking artifacts to each others: Each edge refer to an artifact-artifact relation labelled by its discovered cardinality. The direction of each edge corresponds to that of the causality relation of the two artifacts;

Based on these two figures, we draw some discussions concerning the following points:

- The resolution degree of the discovered artifacts: While some artifacts where discovered at the same resolution degree as that of the manually defined ones (e.g. 'meter_meter',

'contract_transport0contract', 'deal_trade', 'demand0fee', 'book'), others were discovered with higher resolution. Taking the example of BP_1 visualized in Figure 7.16, the discovered artifacts 'zone' and 'cost_price' were separated from the manually defined artifact 'power deal' (which corresponds to the discovered artifact 'hour_hour0end+mw'). As for BP_2, the manually defined artifact 'Gas deal' was divided into three artifacts in Figure 7.15; 'gas_natural0gas', 'volume_volume0mabagement' and 'efective'.
Actually this is due to the fact that employees would not always introduce all the business data values of the same artifacts in emails; they are usually limited to introduce the business data values and types that are manipulated by the discussed activities. As consequence, such dis-matching between our results and the annotated ones could give more precision on the manipulated artifact attributes by BP activities.

- The discovered artifact-artifact relations: Due to the difference between the discovered resolution degree of artifacts and the annotated ones, new artifact-artifact relations are induced by the additional discovered artifacts (e.g. relation between 'effective' and 'meter_mtr' artifacts). Other new relations appear due to the absence of the inheritance relation type in the discovered ones. Taking the example of the artifact 'schedule' in Figure 7.16, it has a relation with the artifact 'hour_hour0end+mw' (which corresponds to what was manually annotated between the artifacts 'power deal' and 'deal' in Figure 7.9) and another relation with the artifact 'deal_trade' (which was absent in the manual annotation but could be deduced from the inheritance relation between 'deal' and 'power deal').

7.7.2 Organizational perspective discovery: Visualization & Example of results

We recall that the discovery of BP organizational perspective mainly relies on speech act discovery which was previously evaluated in the Chapter 6 (see Section 6.5.2 and Section 6.5.3). In this section, we focus on showing some examples of results concerning the organizational perspective discovery and we equally detail how we have visualized them.

One of the main motivations of discovering activity actors in BP is to analyze the underlying social network. In the context of email data, social network analysis can reveal, among others, the communication patterns and relationships between actors. Since our main novelty is the discovery of actors' contributions, a visualization tool that takes them into account was implemented using *networkx* and *matplotlib*. Our implementation provides, for each activity, four different visualizations; three graph types visualizing the interactions between senders and receivers of activity related emails, and a pie chart showing the overall contributions of senders. We illustrate in what follows some examples of the obtained visualizations related to the organizational perspective of the activity *'set_determine_arrange interview{phone}'* (see Figure 7.17, Figure 7.18) and the activity *'set deal{numeric}{ticket}'* (see Figure 7.19 and Figure 7.20). The event sub-logs of these activities correspond to those having the identifiers el_{15} and el_4 in Table 6.4, respectively.

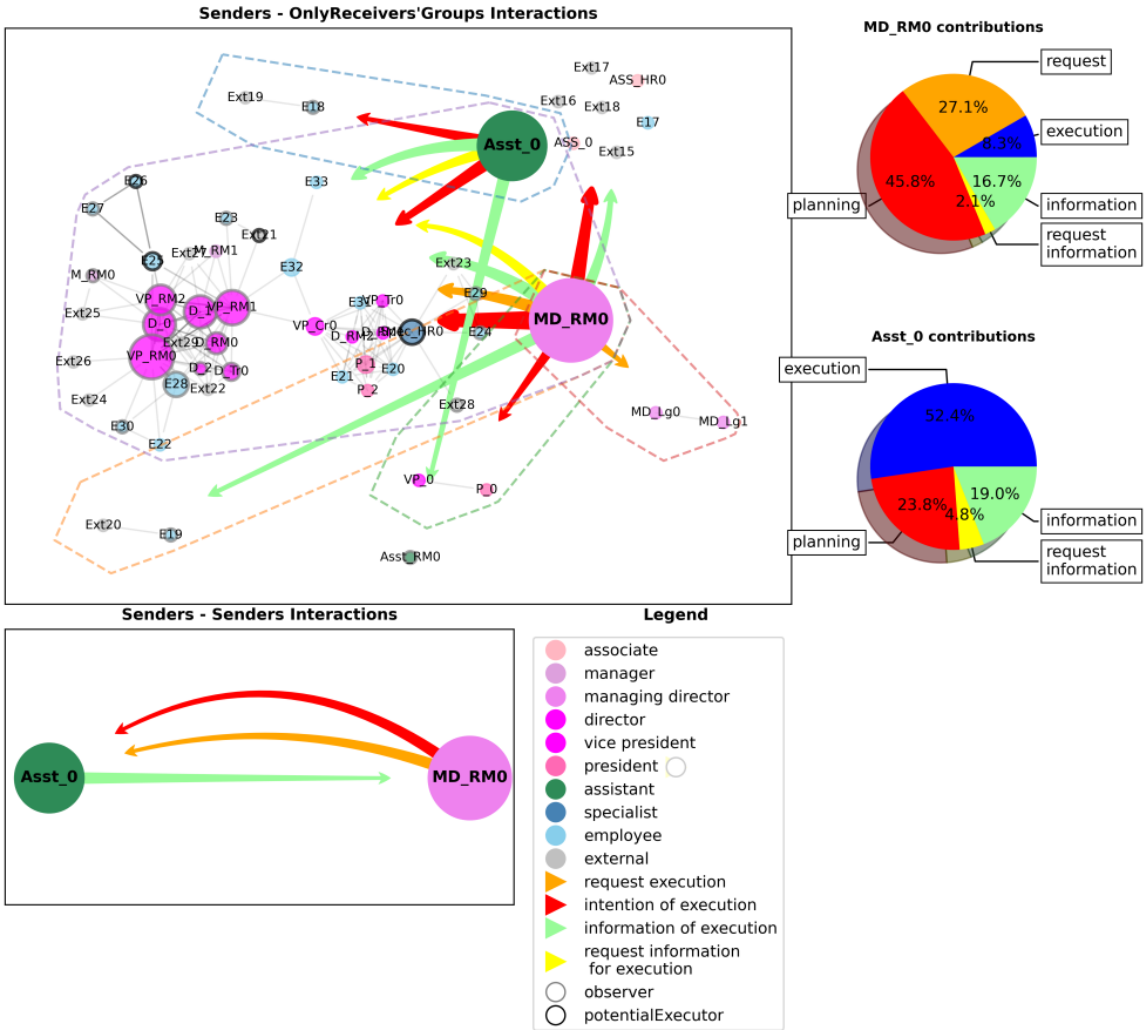
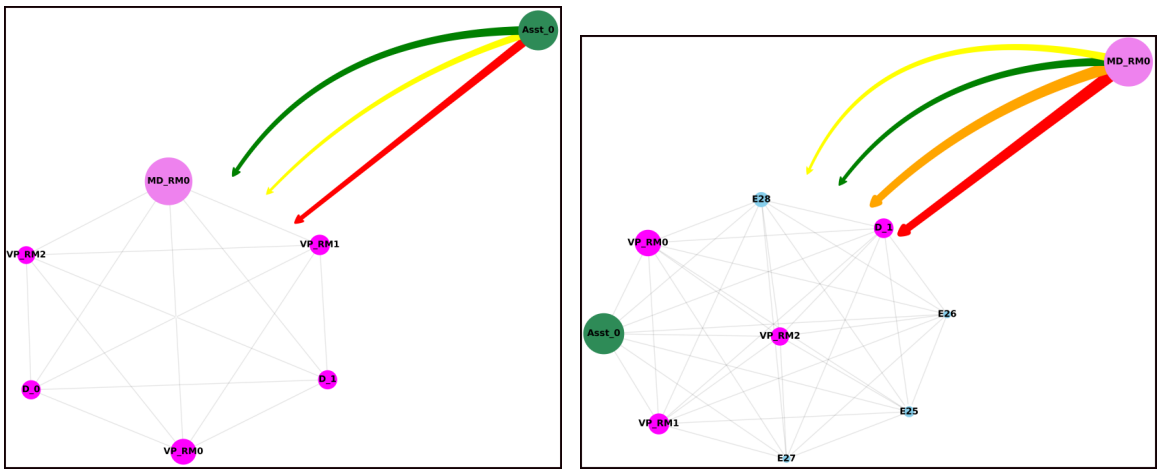


Figure 7.17: Organizational Perspective of the activity *'set_determine_arrange interview'*



(a) Assistant-Receivers Group **(b)** Manager-Receivers Group

Figure 7.18: Sender-Receivers' Groups of the activity *'set_determine_arrange interview'*

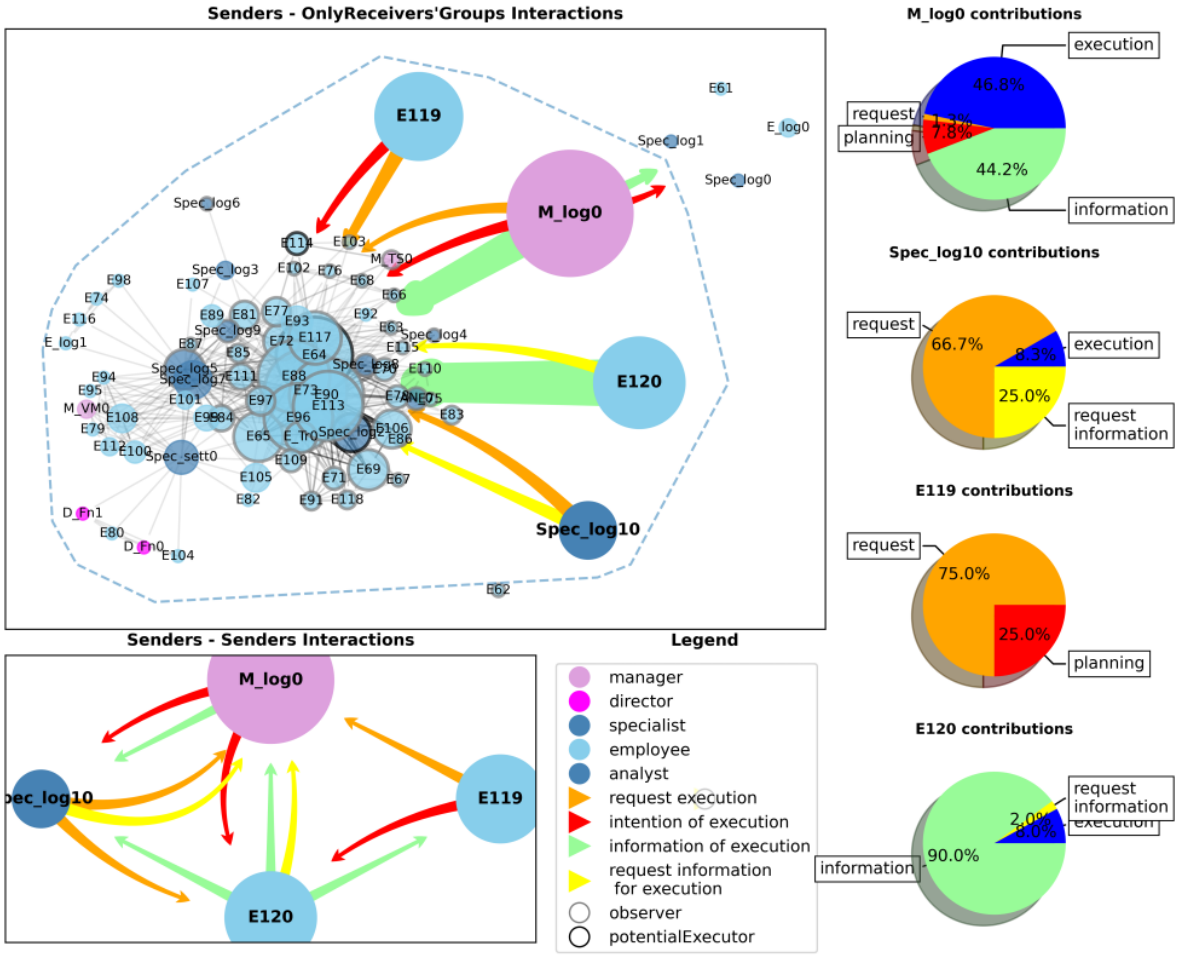
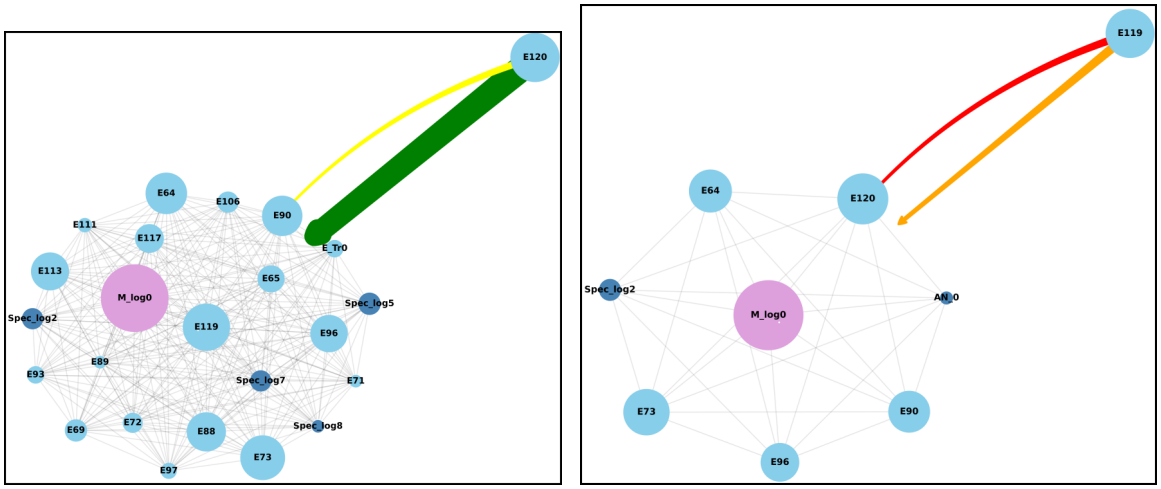


Figure 7.19: Organizational Perspective of the activity *'create deal{numeric}{ticket}'*



(a) E120-Receiver Group **(b)** E119-Receiver Group

Figure 7.20: Sender-Receiver Groups of the activity *'create deal{numeric}{ticket}'*

Nodes in the visualized graphs refer to activity actors. As previously explained, these actors are of two types. The first type corresponds to the senders, i.e. actors that frequently send emails about the activity. We consider in this section that they have to send more than 5 activity related emails. The second type of actors corresponds to the OnlyReceivers who are the actors that only receive emails about the activity.

For each actor node in the graphs, its size is proportional to the number of activity related emails that were sent or/and received by the corresponding actor. Its color refers to the actor organizational role (e.g. the purple color refers to the manager role, see the legends in Figure 7.17 and Figure 7.19). As for its label, it corresponds to the concatenation of: (i) the abbreviation of the actor organizational role (e.g. *'M'* for referring to the manager role), (ii) the abbreviation of his/her business role (e.g. *'log'* for referring to the logistic business role), and (iii) an anonymous ID (e.g. the ID *'0'* for the node label *'M_log0'*). In the case of external actors (i.e. who are not Enron employees), we use the abbreviation *'Ex'*. As for the Enron employees that we do not dispose information concerning their organizational and business roles, we use the abbreviation *'E'*.

Nodes in the graphs with grey (black) edges refer to receivers with an observation (potential execution) contribution. Each colored and directed edge between nodes represents a type of interaction (i.e. a number of emails sent with a specific speech act) between a sender and a receiver or a set of receivers. The edge width is proportional with the corresponding number of emails.

For each activity, we illustrate in its organizational perspective the following types of visualizations:

1. **Graph of Senders-OnlyReceivers interactions** (see Figure 7.17 and Figure 7.19): This graph shows the interactions made by senders and OnlyReceivers. These interactions (i.e. edges) are of two types. The first one is represented by undirected edges indicating that two OnlyReceivers actors were receivers of the same email. A Connected group of nodes through such undirected edges refers to a group of OnlyReceivers receiving the same emails. The second type of edges are directed. Each one refers to a number of emails sent from a sender to a group of OnlyReceivers with a specific speech act. Each closed dashed line in the interaction graph regroups nodes of the same actor group. This means that it comprises the set of nodes belonging to an OnlyReceivers group and all senders interacting with it, e.g. blue and purple lines.

2. **Graph of Senders-Senders interactions**: It summarizes interactions between senders. Taking the example of the node *'E120'* (see Figure 7.19), the corresponding employee always sends information of execution to other senders while receiving intentions or requests. As for the node *'Asst_0'* (see Figure 7.17), it receives intentions or requests of execution from *'MD_RM0'* while sending information of execution to the same node.

3. **Senders' Pie-chart contributions**: It illustrates the fractions in (%) characterizing the distribution of each sender's contributions. e.g. in the context of the creating deal activity, the corresponding employee of the node *'E119'* (Figure 7.19) makes 75%

request and 25% planning of execution.

4. **Graph of Sender-Receivers Group:** It illustrates a sender-receivers group related to one activity (see Figure 7.18 and Figure 7.20). The receiver group and the activity speech acts contained in the emails that they receive from the sender are illustrated in the same way as in the graph of Senders-OnlyReceivers interactions. The main difference is that a receiver group could contain a sender on the contrary of an OnlyReceivers group.

Our visualizations outline the different interactions and contributions made by actors of the same activity (per business role or/and organizational role). Given the case of arranging interview activity (see Figure 7.17), we could understand from the Senders-Senders interactions that the manager (i.e. the node *'MD_RM0'*, which corresponds to the employee **E3** in Table 5.2) always sends requests or intention of execution to his/her assistant (i.e. the node *'Asst_0'*, which corresponds to the employee **E4** in Table 5.2). The assistant contacts then the concerned candidates for scheduling a suitable time slot. Afterwards, he/she goes back to the manager to inform him/her about the fixed time and that the activity was done. The Senders-OnlyReceivers groups and sender-Receivers groups (shown in Figure 7.18) reveal that other actors are involved in these email exchange as receivers without interacting. These actors are: (i) the interviewers that frequently conduct interviews with the manager (e.g. *'VP_RM0'* in Figure 7.17 and Figure 7.18), or (ii) the candidates to be interviewed (e.g. *'Ex15'*, *'Ex16'* and *'Ex17'* in Figure 7.17).

The visualization would additionally provide more understanding regarding activity execution rights. Given the example of the employees *'Spec_log10'* and *'E119'* (see Figure 7.19, Senders-Senders interactions), it's clear that they do not have the right of creating a deal (through an information system). This is why they always contact a manager (*'M_log0'*) or a specialist (*'E120'*) having access. This would recommend; (i) who to be contacted for executing an activity or for intervening in case of a delayed activity, and (ii) extensions to be implemented in an information system to further automate actors' interactions.

7.7.3 Behavioral perspective discovery

To assess the results of our behavioral perspective discovery step, we evaluate the generated sequencing constraints in terms of relevance. In what follows, we explain our annotation methodology and we discuss the results of the experiments that we carried out.

7.7.3.1 Annotation Methodology

To annotate sequencing constraints in terms of relevance, emails or/and successive emails that have induced the appearance of constraint candidates were first exported. Then, for each one, each occurrence was annotated in terms of significance. A constraint candidate occurrence was considered significant if:

- The reference and the target events handle the same instance (i.e. relevant information value); AND

- There is a conditional relation between the reference and target events (i.e. the reference event has induced the occurrence of the target event)

One sequencing constraint is considered relevant if more than half of its occurrences were considered significant.

7.7.3.2 Experiments and Results

Our step that generates sequencing constraints depends on two main parameters when selecting the final constraints: (i) $th_{N_{constOcc}}$: the minimum number of constraint occurrence $N_{constOcc}$, and (ii) $th_{Confd_{const}}$: the minimum constraint confidence value $Confd_{const}$. For studying these two thresholds parameters, we show in Figure 7.21 the distribution of the number of relevant and non relevant constraint candidates according to $N_{constOcc}$ and $Confd_{const}$. Bar lengths in the figure reflect the total number of constraints verifying a couple of parameter values. These values vary from 3 to 43 in the case of number of occurrence parameter and from 0.2 to 1 in the case of confidence measure parameter (after rounding its values to tenths). The orange color of each bar reflects the ratio of relevant constraints from the overall ones verifying the related parameter values.

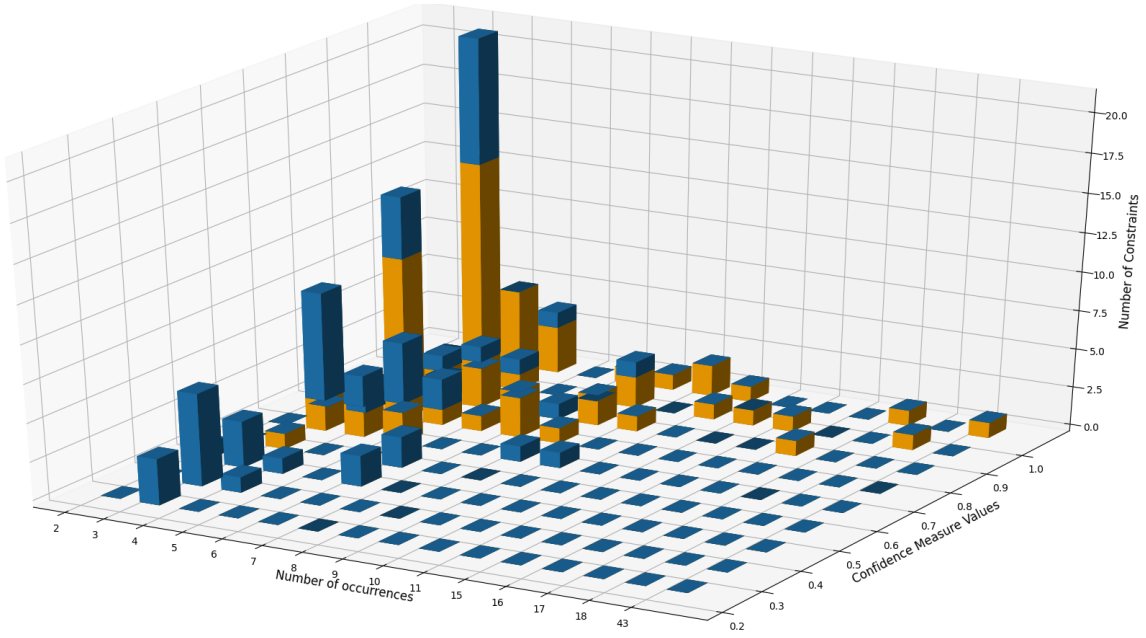


Figure 7.21: Number of Constraints Distribution Per Confidence Measures and Number of Occurrences

Based on the obtained distribution, we underline the following observations:

(1) For low confidence values (i.e. $Confd_{const}$ is close to 0): Constraints appearing at this level would be potentially non-significant whatever the number of their occurrences. Let consider one constraint $const_{ij}$, formed by a reference event e_i and a target event e_j and having a low confidence value $Confd_{ij}$. This means that a contradictory constraint $const_{ji}$ of a reference e_j and a target e_i exists with high confidence value $Confd_{const} = 1 - Confd_{ij}$. Additionally, this indicates that $const_{ji}$ has logically occurred much more than $const_{ij}$ that appeared at low number of occurrence (compared to $const_{ji}$). Actually, such low number of occurrence would be explained by: (i) false detection of events e_i and/or e_j, or (ii) successive appearance of e_i and e_j but in different contexts (e.g. different instances) and without a conditional relation between them.

(2) For high confidence values (i.e. $Confd_{const}$ is close to 1): The obtained results show that high confidence measure is not sufficient to obtain relevant constraints. Indeed, for a maximum confidence value $Confd_{const} = 1$, we obtain an important number of non relevant constraints (i.e..) if $N_{constOcc} = 3$. When increasing $N_{constOcc}$, the number of relevant constraints will be dominant over the non relevant ones. Meanwhile, we could skip the detection of some significant constraints if $th_{N_{constOcc}}$ is of very high values (e.g. for $N_{constOcc} \in [17, 43]$, all relevant constraints of lower number of occurrence are to be ignored).
It is important to note that the higher the $N_{constOcc}$, the closer the confidence measure is to 1 and the less the constraints of low confidence measure appear (e.g. for $N_{constOcc} > 8$, there are no constraints that were detected having confidence measure lower than 0.5).

To conclude, both $th_{Confd_{const}}$ and $th_{N_{constOcc}}$ could influence the relevance degree of the discovered response constraints. $th_{Confd_{const}}$ must be logically ≥ 0.5 to avoid the case described in (1). As for $th_{N_{constOcc}}$, low values would be preferably avoided (e.g. 1, 2, 3) as they potentially correspond to combination of events wrongly detected or randomly appeared in successive appearance order (in emails or threads).

7.7.3.3 Visualization and Discussion

To illustrate the obtained results, a visualization tool was implemented using the *pydot*[2] graph library. Figure 7.23 and Figure 7.22 (a) show the Activity models obtained for BP_1 and BP_2. Figure 7.24 (b) and Figure 7.22 (b) show examples of Activity & SA models obtained for the BP fragment BP_3 and BP_2. These examples were generated by setting $th_{Confd_{const}} = 0.5$, $th_{N_{constOcc}} = 4$ and $\alpha = \gamma = 0.4$ (we only consider $th_{N_{constOcc}} = 3$ at the level of the Activity & SA model of the BP fragment BP_3 as we noticed that the corresponding sequencing constraints do not have high numbers of occurrences).

Each generated model provides a graphical organization for *response* constraints. It has the form of a directed graph while each directed edge linking a parent and a child node refers to a reference and a target event of the same constraint. In this way, actions that could be taken as a response to a given event will have directed edges linking them to the same

[2]https://pypi.org/project/pydot/

reference/parent node. Nevertheless, it is important to note that nodes located at the same level do not necessary refer to activities simultaneously performed . Taking the example of *'expire deal'* and *'flow deal'* in Figure 7.22 (a), they do not refer to events having temporary constraints (e.g. they have to start at the same time).

The colors of the nodes in the visualized models are in relation to the speech acts of the event types. For the graphs referring to activity models (e.g. Figure 7.23 and Figure 7.22 (a)), each node can be colored using two colors: (i) a green color to reflect an information speech act (i.e. referring to a past tense), and (ii) a pink color to reflect the other speech acts (i.e. *'request'*, *'intention'* and *'request information'* speech acts which would potentially refer to a future tense). The part that each color occupies reflects the percentage with which the events verifying the corresponding constraint have appeared with each speech act category. As for the graphs referring to activity & SA models (e.g. Figure 7.24 (b) and Figure 7.22 (b)), the node colors refers to the speech of the corresponding event types, i.e. the green, orange, purple and yellow colors refer to the *'information*, *'request'*, *'intention'* and *'request information'* speech acts, respectively.

In these figures, we highlighted with red colour directed edges referring to sequential constraints that were manually defined as non-significant ones. This would give insights into the relevance degree of the generated models. Given the example of Figure 7.22 (b), 3 from 11 constraints were identified as non-relevant which induces a relevance ratio of 0.72 for the overall model. As for the other model examples, the related relevance ratios correspond to 0.66 (in Figure 7.22 (a)), 0.6 (in Figure 7.23), 0.8 (in Figure 7.24 (a)) and 1 (in Figure 7.24 (b)).

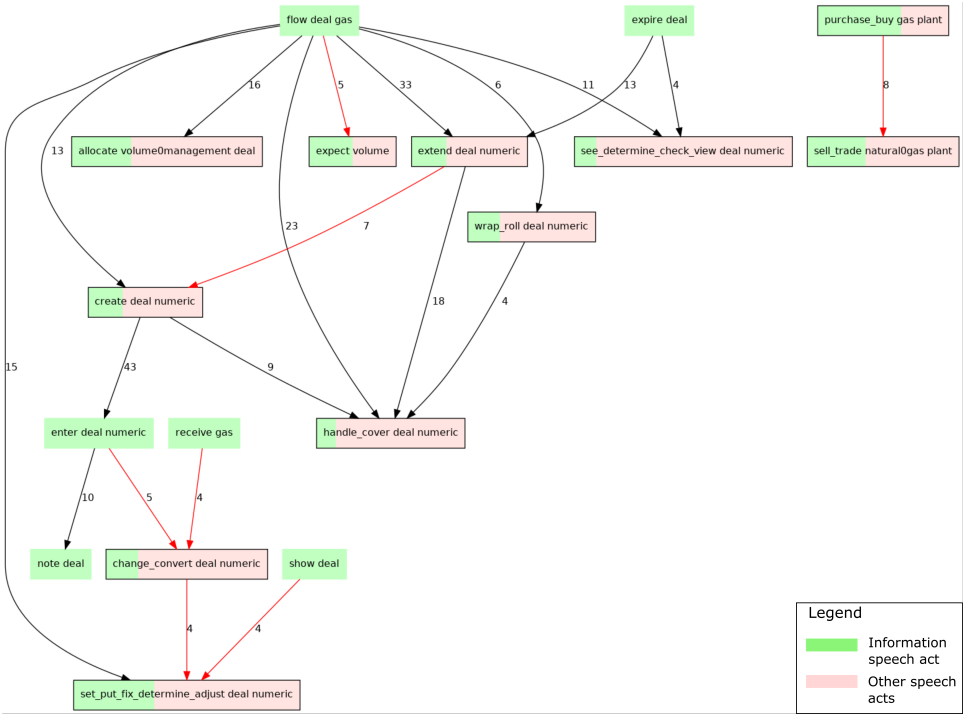
Based on these models, we could understand the behavioral perspective of the BP fragment:

(i) For trading electricity power BP fragment (BP_1): We could understand from the activity model in Figure 7.23 that following an information concerning a trading activity (i.e. opening a long purchase power position or purchasing power), the employees inform about its scheduling in the trading information system. Additionally, the employees use emails to inform about opening and scheduling a long trading position of purchase. Then, they express their intention to sell power to the organism having opening these positions. Furthermore, following a cut of a deal, they replace it by another one;

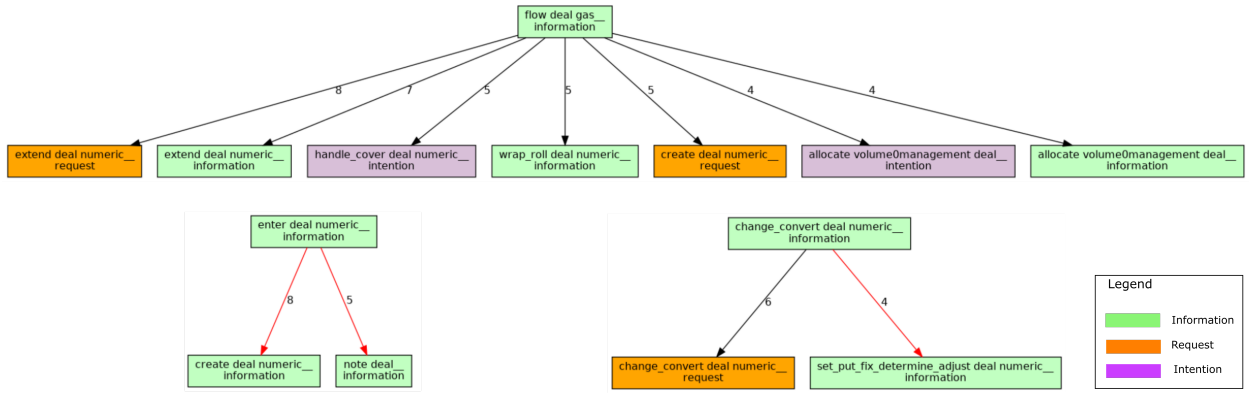
(ii) For managing gas deal BP fragment (BP_2): The models in Figure 7.22 shows that two main event types generally initiate interactions between gas trading actors: (i) *'flow deal gas'* , and (ii) *'expire deal'*. Various actions are then performed following these events. There are those who are in common, namely extending or checking the corresponding deal. Others that are performed to handle the flow gas event such as: create a new deal, allocate additional gas volume and roll the corresponding deal;

(iii) For organizing interview BP fragment (BP_3): Figure 7.24 (b) shows that following an information concerning forwarding a resume of a candidate or requesting arranging an interview (which is generally performed by the manager according to the corresponded actor perspective), the receiver (who is generally the assistant) sends an email

to the concerned candidate to express an intention of organizing an interview with him/her. Then, he/she goes back to his/her manager to inform him/her about the scheduling/rescheduling of the interview;



(a) Activity Model of BP_2



(b) Activity & SA Model of BP_2

Figure 7.22: Behavioral Models of Trading Gas BP fragment (BP_2)

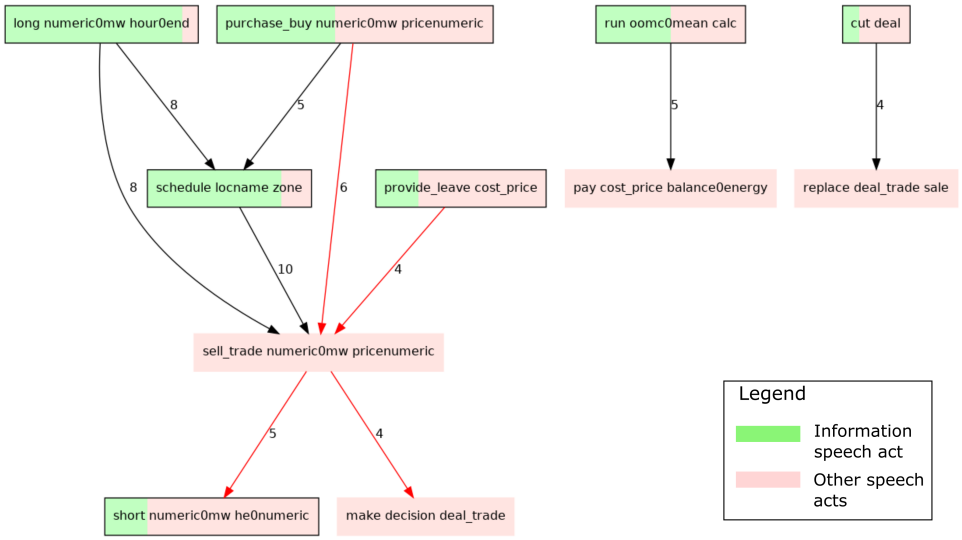
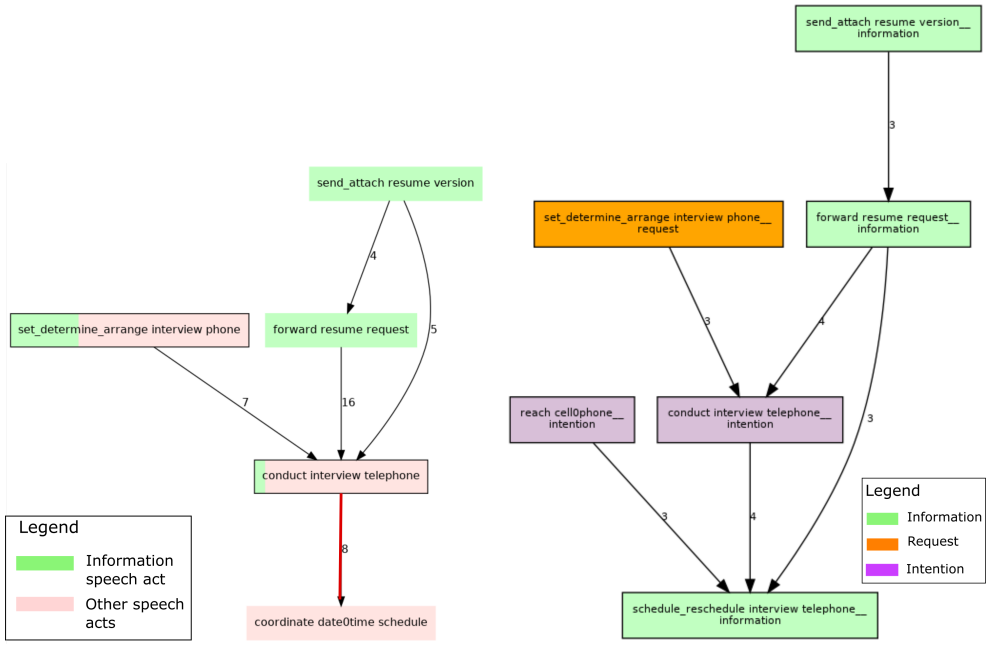


Figure 7.23: Behavioral Perspective Model of Trading Power BP fragment (BP_1)



(a) Activity Model of BP_3 **(b)** Activity & SA Model of BP_3

Figure 7.24: Behavioral Models of organizing interview BP fragment (BP_3)

7.8 Conclusion

In this chapter, we achieved the second part of our second objective (**Objective 2.2:** Automate the discovery of the defined BP knowledge from emails). We equally answered the second question (**Q2**) raised in the thesis problematic (Section 1.2.2), which is : How to discover BP knowledge from emails? according to the following sub-questions:

- Q2-1: How to discover BP elements without disposing a priori knowledge about them ?
- Q2-5: How to deal with the uncertainty of BP element presence in emails (e.g for estimating the relative order of events in the absence of their exact timestamps) ?
- Q2-6: How to mine the generated structured event log to discover multiple BP perspectives ?

To this end, we have proposed an unsupervised approach based on these key features:

- It applies overlapping clustering on BP elements to discover artifacts and BP fragments. This leads to designate one BP element to more than one artifact or more than one BP fragment, which is adequate to reality;
- It exploits activity occurrences' speech acts to deduce how emails are used in the context of BP. This enables to: (i) enrich the behavioral perspective through event types combining activities and their speech acts, (ii) approximate the relative sequencing order of activity events in the same emails/threads, and (iii) enrich the BP organizational perspective through the notion of actor contribution and types of interactions between actors;

Our experiments were carried out on a public dataset and the results are publicly shared (see the *Event Log Mining* section in this link[3]), which is, in our knowledge, absent in related studies (This is why comparison with them was not feasible when evaluating our proposals). The obtained results show good performances with respect to emails of the selected Enron employees.

We agree that our approach needs some parameters to be set by the user (e.g. activity similarity threshold α for artifact discovery, confidence measure thresholds γ and β for building causality relations used for BP fragment discovery). However, we showed through our experiments that these parameters could be logically set (e.g. in the neighbourhood of 0.5 for α, γ and β) without having to vary them for very high or very low values. Additionally, this could be an alternative for being able to discover BP elements without prior knowledge or having to provide the number of categories for each BP element.
We agree also that further analysis must be carried out to study if our main assumptions

[3]http://www-inf.it-sudparis.eu/SIMBAD/tools/processDiscoveryFromEmails/

remain valid while covering larger set of employees and emails of other companies. Moreover, additional experiments must be conducted especially for quantifying the recall of the discovered behavioral perspective models. This was absent in our work due to the lack of Enron BP documentation and the lack of business expertise concerning how they are supported in emails. Finally, we must further investigate some algorithmic assumptions used to approximate the discovery of some BP elements related to:

- The instance identifier; actually, an approximation through a combination of information types values must be studied, instead of relying on a single value of a single information type;
- The occurrence of one artifact; it was approximated in our work by the occurrence of one of its attributes. Indeed, this is limited as two artifacts could share some attributes;

This work paves the way for smart features in email and BP environments. In the next chapter, we will present two potential applications (Chapter 8) that extend the obtained results in order to enhance BP management.

Chapter 8

BP discovery from emails: Potential Applications

Contents

8.1 Introduction

Up to now, we have introduced all techniques of our overall approach for BP discovery w.r.t multiple perspectives. In this chapter, we show how these techniques could be extended to new fields that are different from BP discovery but that improve BPM. Towards this goal, we present two potential applications of:

1. BP discovery & recommendation tool to be integrated in email clients. We show through this application the utility of BP discovery for improving BP knowledge management in emails;

2. CRM (Customer Relationship Management) data analysis for discovering reasons of users' satisfaction and non satisfaction. We show through this application how our activity discovery solution could be applied to other types of textual data other than emails.

This chapter is organized as follows. In Section 8.2, we present our first potential application. We show the usefulness of our obtained results for conceiving a BP discovery & recommendation system to be integrated in emailing systems. This recommendation system aims to assist employees when performing BP activities within emails by providing them relevant BP knowledge. It is composed of two main components. The first component aims to ensure an initialization phase for BP discovery from emails. We present in its context our vision concerning its implementation architecture for ensuring BP discovery from emails while respecting confidentially constraints. The second component aims to exploit the discovered BP knowledge from the first component for providing useful recommendations for employees. In Section 8.3, we talk about the second potential application where we extend our activity discovery solution for analysing CRM (user Relationship Management) data. The main goal of this application is to mine reasons of satisfaction and non-satisfaction of users of organisms' services. We use real-life dataset retrieved from a telecom operator CRM data to illustrate examples of results and we discuss potential use cases. Finally, Section 8.4 summarizes this chapter.

8.2 BP discovery & Recommendation System

In this section, we focus on a the first potential application to show the usefulness of our results for conceiving a BP discovery & recommendation system to be integrated in emailing systems.

Generally, the ultimate goal of BP discovery from execution logs is to improve BPM. Nevertheless, up until now, existing studies that discovered BP from emails were limited to associating to each email the set of BP elements included in it. Tools that were designed in the same context allow employees at the most managing ongoing activities, e.g. by summarizing activities included in received emails [77] or displaying activity realization status [25]. Unfortunately, email traces remains unmanaged w.r.t BP. Therefore, we aim from the BP discovery & recommendation system integrated in email clients to fill the gap between BP discovery from emails and BPM.

The global architecture of the BP discovery & recommendation system that we propose is illustrated in Figure 8.1. It is composed of a centralized server and a plugin installed in the email client of several employees, which mainly ensure two phases:

(i) The first phase is an initialization phase. Its goal is to analyze employee emails in order to discover BP w.r.t multiple perspectives. The main challenge is how to overcome confidentiality issue related to the analysis of personal data as the case of employee emails. We explain in Section 8.2.1 our vision concerning its implementation architecture while respecting confidentiality constraints;

(ii) The second phase is a recommendation system that exploits the knowledge extracted in the initialization phase to generate recommendations for employees. We detail the

architecture and the implementation of this phase in Section 8.2.2. We show additionally some example of results;

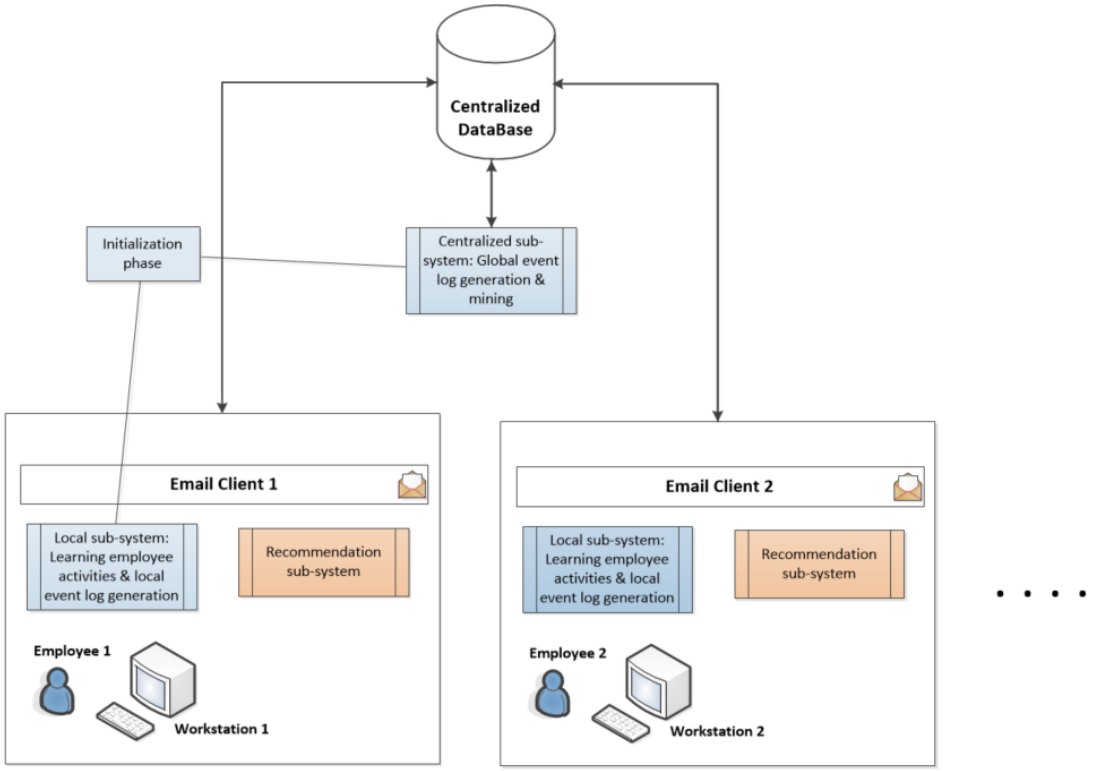


Figure 8.1: Global architecture of BP recommendation system integrated in email clients

8.2.1 Initialization Phase & Confidentiality issue

Textual contents of emails (especially those related to their main bodies) are considered as personal and sensitive data with regards to employees. Therefore, collecting emails of different employees for applying any type of analysis requires their permission. Actually, employees rarely agree to share the textual content of their emails to centralize their access. Hence, it would not be always feasible to rely on a centralized analysis of emails where those of different employees are simultaneously processed.

We explain in what follows our vision to overcome such privacy issue of emails by enabling a distributed architecture of email analysis for BP discovery (w.r.t employees). As illustrated in Figure 8.1, we propose to distribute the step of learning activities and event log generation from emails on employee mail boxes (i.e. *Local sub-system*). This means that emails are to be locally analysed in each employee's machine. The goal is to allow each employee to intervene after the step of activity and patterns discovery to: (i) discard those that are judged confidential for him/her, and (ii) validate sharing the remained ones. Afterwards, we use such validation results to generate a local event log at the level of each employee's machine to be shared and stored in a centralized database. Sensitive data in each local event logs are:

(i) anonymized in the case of email textual contents, and (ii) hashed in the case of relevant information values. The overall local event logs are then to be handled by a *Centralized sub-system* to be merged into a global event log allowing its mining for BP discovery w.r.t. multiple perspectives. We detail in what follows the main steps composing the phase of anonymisation & hashing for local event log generation:

- **Email content Anonymization:** This step generates an anonymous version of each email to guarantee the possibility of centralizing their analysis. It reduces the main body of each email to the patterns that it contains and then to the set of previously validated activity labels (in terms of sharing). This actually corresponds to the step of activity occurrence discovery (Step 2.2).
- **Local thread construction:** At the level of each employee mailbox, this step constructs local threads. Each local thread actually presents a partial view of a global thread that could be constructed if emails of different employees are analysed. That's why, during local thread construction, we preserve a set of features enabling each thread partial view to be linked to the other views of the same global thread at the level of the centralized sub-system. Such features refer to the emails received from other collaborators and belonging to the same local threads.
  To this end, at the level of each employee mail box, we first locally constructs threads from all sent and received emails. We remind that our step of constructing email threads (Step 2.3) relies on grouping email conversations sharing the same relevant information values (i.e. approximating instance notion). That is why, for each employee, these values are to be locally identified in his/her mailbox. Afterwards, for each thread, emails that are not sent by the employee are removed after keeping the following features: timestamp, sender, receivers and the type of relations with the emails of the thread to which it belongs. This relation could be conversational (i.e. reply or forward to an email) or due to a shared relevant information value with the thread emails.
- **Hashing local threads features for sharing purposes:** We remind that local thread features correspond to: (i) email addresses of received email senders, or/and (ii) relevant information values. Such type of information actually refers to confidential data that are not allowed to be shared at the level of an employee mail box. Since these data presents a way to map the different local views of the same threads, we don't really need their real values. It is enough to turn each of them into a kind of unique code to be shared rather disclosing their actual contents. To this end, we propose to transform them into hash codes using a hashing algorithm (e.g. MD5, SHA256). Hashing is the process of converting the information into a key/code. The original information cannot be retrieved from the hash code by any means.

8.2.2 Recommendation Phase

In this section, we describe the architecture of the recommendation sub-system. This sub-system takes as input the discovery results of the initialization phase. It provides BP oriented

recommendations for email users when performing their activities. Generally, recommendations that would be provided in emailing systems could be in two modes: (i) read mode, which means when reading a received email, and (ii) compose mode, which means when composing an email to be sent. In this work, we focus on providing recommendations in read mode. More precisely, we aim to provide two types of recommendations when an email user receives a BP related email:

(i) Graphical recommendation: It displays, through graphs, in the context of which current activity/activities the received email is inscribed. It equally illustrates the next recommended activities to be performed by the user as a response;

(ii) Textual recommendation: It provides textual suggestions in relation to the following elements: (i) the list of business data values in relation to the current and next activities, and (ii) the textual email contents to be sent when performing the next activities;

We illustrate in Figure 8.2 the main steps describing how the recommendation subsystem works to generate such type of outputs. We detail in what follows the implemented functionalities per each step:

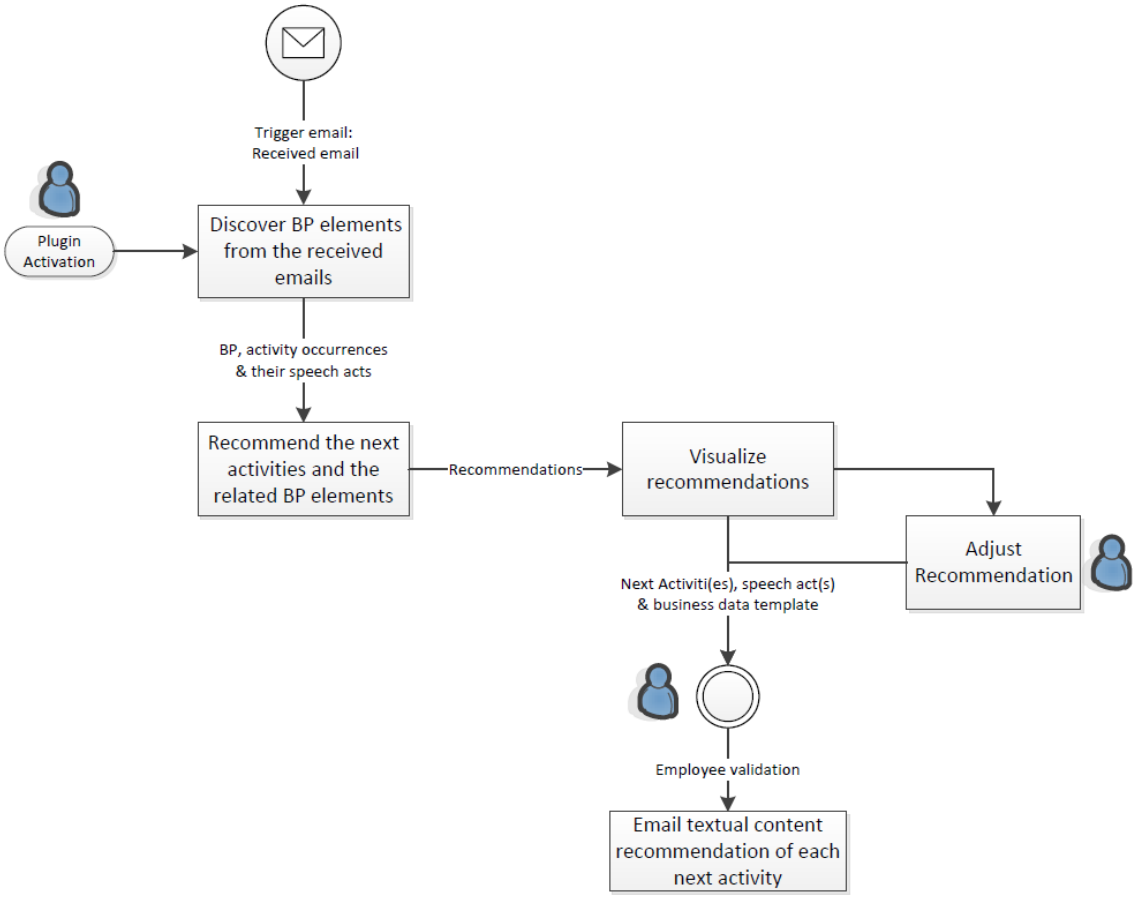


Figure 8.2: Recommendation sub-system

Step 1: Discover BP elements from the received email: This functionality is initiated when a triggering email is received. It allows to detect all the occurrences of the activities present in the received email including their business data components and the corresponding speech acts. Concretely, it applies our solution for activity occurrences discovery. It uses as input the triggering email and the results of the initialization phase in terms of the discovered activities and BP. To identify to which BP fragment the received email belongs, we use a voting approach where each discovered BP element votes to the BP fragment(s) to which it was assigned in the initialization phase. We finally pick the BP fragment with the highest number of votes.
Step 2: Recommend the next activities and the related BP elements: This step uses the discovered BP elements in the received email to recommend the next activities, the corresponding speech acts, business data template and values. To recommend the next activities/speech acts, we rely on the results related to the BP fragment behavioral perspective discovered at the level of the event log mining phase. For each current activity occurrence associated to its speech act, we retrieve all the sequencing constraints having it as a reference event type. Then, we select the corresponding target event types that we use for inferring the recommended next activities.
For recommending the business data template of each next activity, we rely on the attributes of its artifacts discovered in the initialization phase. In other words, the business data template is formed by the set of artifact attributes. As for recommending the values of each attribute, we search its occurrences in the received email to retrieve the corresponding values. However, the occurrence of the artifact attributes of a next activity in a received email is not always guaranteed. Therefore, for each attribute, the recommendation sub-system returns a set of values that could be: (a) empty where the user will be invited later to fill its value(s), or (b) non-empty where the sub-system provides its elements as suggestions to the user;
Step 3: Visualize & Adjust recommendations: In this step, the recommendation sub-system provides the user with graphs that describes: (ii) the BP fragment in relation to the received email, and (ii) illustrations on the current activities (i.e. discovered in the received email), the next activities (i.e. to be performed following the reception of the email) and the corresponding speech acts. It equally provides a first part of the textual recommendation related to the business data template and values of the next activities.
The recommendation sub-system allows the user to interacts with the graphical and the textual recommendation results. The user could directly validate the proposed recommendations. Otherwise, he/she could intervene to make adjustments by: (i) picking another next activity, or/and (ii) adjusting the set and the values of the recommended business data.
Step 5: Email textual content recommendation of each next activity: The validation of the previous recommendations triggers the generation of an email main body to be suggested for performing each next activity. To this end, we assume that the email main body used in the context of performing an activity is the concatenation of invariant and template sentences. These sentences are as follows:

(i) Invariant salutation sentence at the beginning of the email, e.g. *'Good morning'*, *'Hi'*, etc;

(ii) Invariant sentence rephrasing the activity in accordance to its speech act, e.g. the

sentence *'We would conduct an interview with you'* rephrases the activity *'conduct interview'* in accordance to the speech act *'intention'*;

(iii) Template sentences where each one is related to one activity business data attribute. A template sentence is used by the recommendation sub-system to: (a) include the recommended values in the generated email, or (b) allow the user to express the corresponding values in the case of a business data attribute with an empty set of recommended values, e.g. for the business data attribute referring to the interview date, the related sentence template would be *'The interview will be held on datenumeric'*. As shown in the example, a template sentence includes an expression for including the attribute value through a variable tag that corresponds to a numeric (e.g. *'datenumeric'*) or a named entity tag (e.g. *'personname'*);

(iv) Invariant sentence at the end of the email which would salutation and user signature, e.g. *'Sincerely,*
Shirley Crenshaw
Administrative Assistant';

We also assume that that each invariant or template sentence is to be obtained from the initialization phase. For instance, for each activity component, we dispose all the sentences where it occurs. Additionally, we retrieve for each activity the sentences that begin and end the emails where it appears. In this way, for a given email user, we retrieve the set of sentences of the same type (i.e. salutation sentences, ending sentence, and sentences expressing the same business data attribute or activity name). We replace numeric and named entities by tags. Then, we pick one representative sentence for each type by mining the most frequent one or sequence of words.

8.2.3 Example of implementation

In this section, we focus on an example of implementation of the BP discovery & recommendation system. More precisely, we describe the integration format in email clients and the development tools used for implementing the recommendation phase. Then, we illustrate an example of execution of the implemented recommendation sub-system.

8.2.3.1 Implementation

Generally, employees do not prefer changing their working practices or the adoption of new tools when performing their daily business activities [53]. That is why, we chose to implement the recommendation sub-system as a plugin installed in an email client. In this way, employees could benefit from the recommendation functionalities when using their email clients without the need to open a new tool. For the email client where the recommendation sub-system is to be integrated, we have chosen to use the Outlook[1] email client.

[1]https://outlook.live.com/owa/

To develop and integrate a plugin into Microsoft office products, it could be of three types: Component Object Model[2] (COM), Visual Studio Tool for Office[3] (VSTO) and Office Web Add-ins[4]. In our case, we have adopted the Office Web Add-ins plugin due to its flexibility of integration in different operating systems (Windows, Mac Os, IOS, Android). The figure below (Figure 8.3) shows the two basic components of this type of plugin:

- Web application: It is the body of the plugin. We developed it using the angular framework including a set of web development tools (i.e. *html, css, JavaScript, TypeScript*), and the *Node.js*[5] as a technology for the server-side scripting[6]. To interact with the Outlook email client where the plugin is installed, we additionally relied on the *Office.js*[7] wich is an Java Script API (Application Programming Interface).
- Manifest: It is an XML (Extensible Markup Language) file which specifies the parameters and the functionalities of the plugin, namely: (1) The displayed name, description, ID, version, and default regional settings of the add-in, (2) How the add-in integrates with Office, and (3) The authorization level and data access requirements for the add-in.

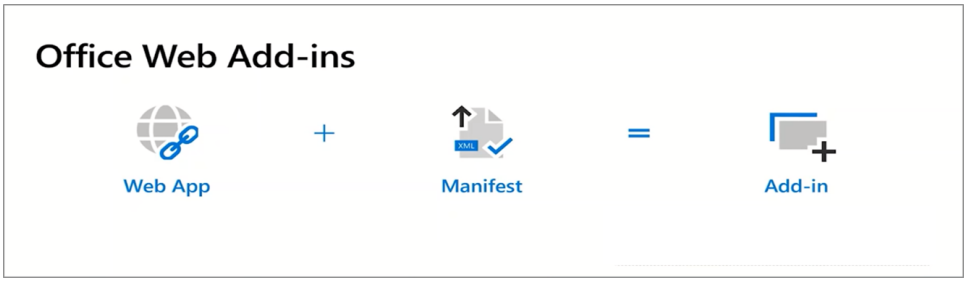


Figure 8.3: office-web-addin components

8.2.3.2 Example of Scenario of use

We show in this part an example of a scenario where the plugin is used in email reading mode. To this end, we install the plugin in the Outlook client of an employee. We assume that this plugin accesses the results of the initialization phase applied on the Enron email dataset that we have analysed in the previous chapters (Chapter 5 & Chapter 7). Then, we send to this employee a fake email in the way that it belongs to the organizing interview BP fragment (BP_3). This email is sent by the manager (i.e. *'Vince*) to his assistant (i.e. *'Shirley'*). He asks her to arrange an interview with a candidate named *'Prakash'*. We illustrate in Figure 8.4 the received email and the displayed results in this scenario. When the administrative assistant

[2]https://docs.microsoft.com/en-us/office/vba/outlook/concepts/getting-started/support-for-com-add-ins
[3]https://en.wikipedia.org/wiki/Visual_Studio_Tools_for_Office
[4]https://docs.microsoft.com/en-us/office/dev/add-ins/overview/office-add-ins
[5]https://nodejs.org/en/download/
[6]The server-side scripting consists of running scripts server-side to produce dynamic web page contents before the page is sent to the user's web browser
[7]https://docs.microsoft.com/fr-fr/office/dev/add-ins/reference/javascript-api-for-office

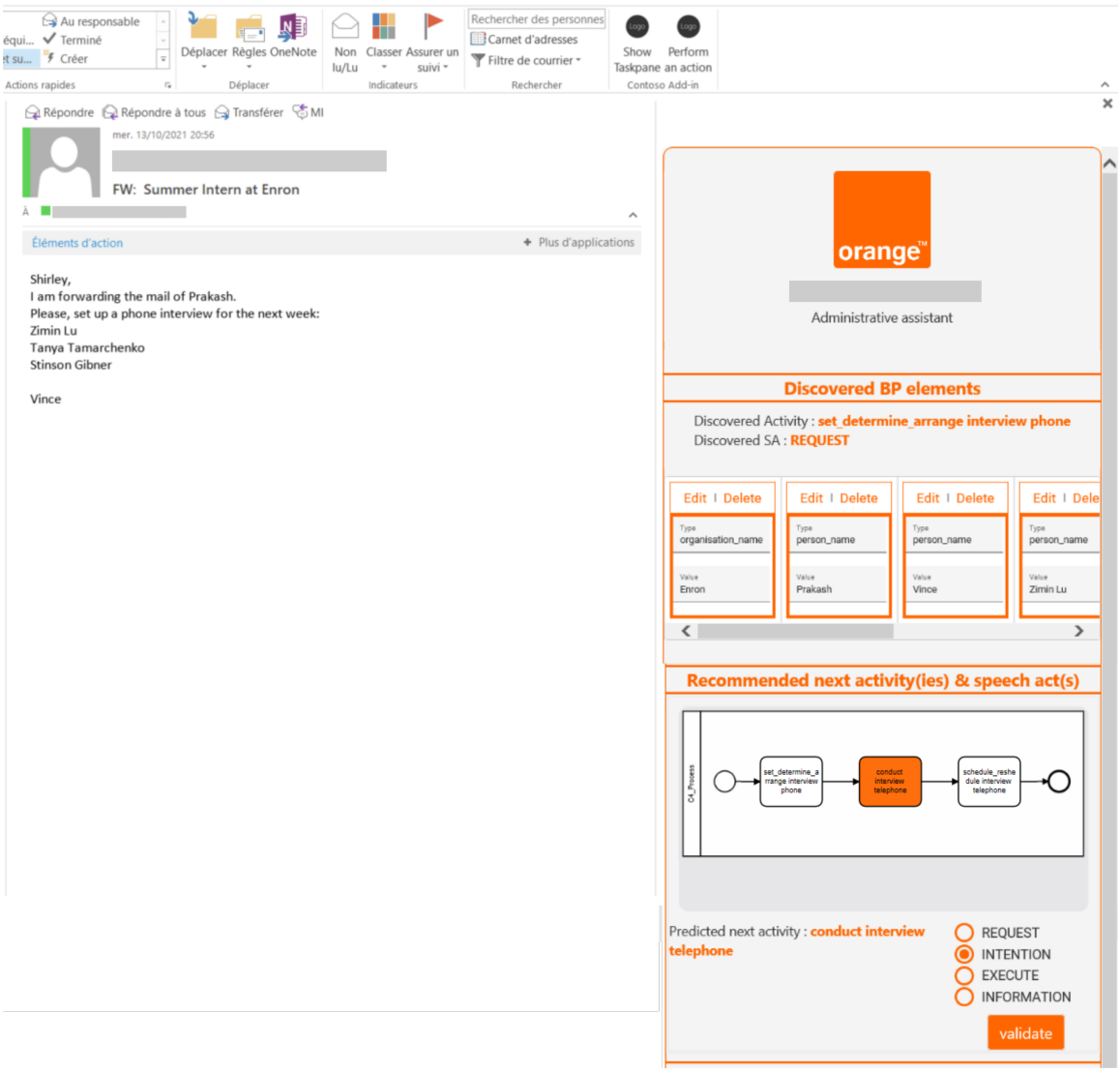


Figure 8.4: Plugin in read mode: Part 1

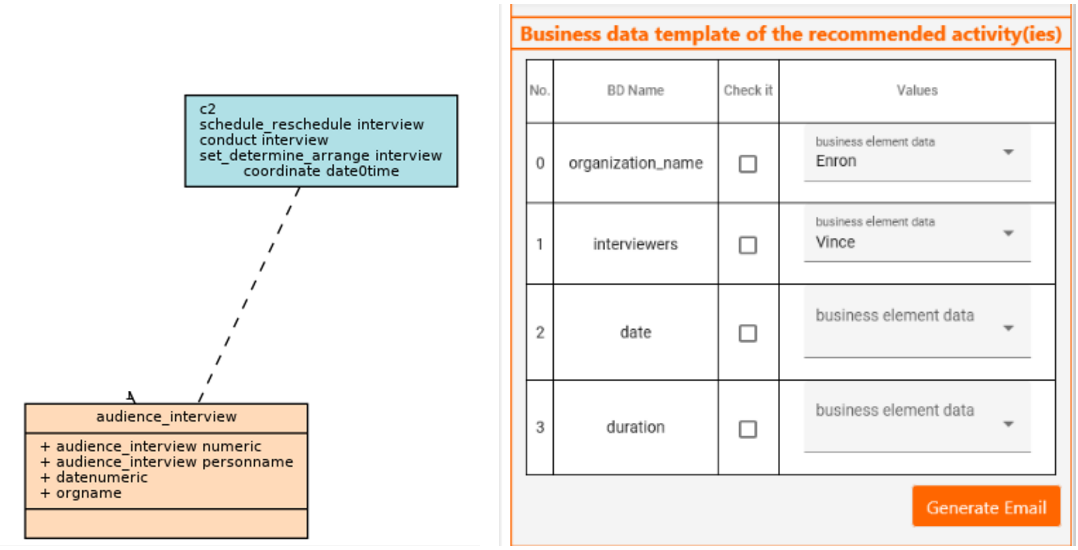
(i.e. the employee that which simulates the scenario) opens the received email and launches the plugin by clicking on *Show Taskpane* icon, the plug-in analyses the email. Then it displays three sections:

(1) Employee Info: The plugin displays the employee name and organizational role within the company;

(2) Discovered BP elements: The plugin displays the names of the discovered activity(ies) from the received email as well as the corresponding speech act(s), i.e. arrange interview (*'set_determine_arrange interview'*) with the speech act *'request'*. It additionally shows the discovered business data values, i.e. *'Enron'* as organization name and *'Prakash'*, *'Zimin Lu'*, *'Tanya Tamarchenko'* and *'Stinson Gibner'* as person names (see the plugin section *Discovered BP elements* in Figure 8.4).

(3) Recommended next activity(ies) & Speech act(s): The plugin displays, through a graph, the part of the BP fragment (i.e. organizing interviews) to which the discovered activity belongs. It displays the discovered activity in the received email (i.e. *'set_determine_arrange interview'*). It equally highlights the recommended next activity to be performed and according to which speech act(i.e. *'conduct interview telephone'* w.r.t *'intention'* speech act). It is important to note that the plugin focuses on the sequencing constraints related to the discovered and the recommended activities to show the relevant BP fragment part to the employee. In the case of the considered email in this example, the plugin rely on the sequencing constraints discovered in the event log mining phase and that concern the organizing interview BP fragment (see the previous chapter, i.e. Chapter 7), namely the sequencing constraints resumed in Figure 7.24 (b).

Once validating/adjusting the recommended next activity, the plugin retrieves its business data template inferred from its artifacts' attributes. Figure 8.5 shows: (i) the artifact attributes as discovered in the initialization phase (Figure 8.5 (a)), which are mainly related to the discovered artifact *'audience_interview'*, and (ii) the displayed business data template by the plugin after relabeling them at the initialization phase level to be more interpretative. The displayed business data template by the plugin lists four business data attributes:



(a) Data perspective extract of organizing interviews

(b) Recommended business data template & values

Figure 8.5: Discovered artifact attributes vs recommended business data template

(i) Organization name (*'organization _name'*) which is equivalent to the artifact attribute *'orgname'*:

(ii) Interviewers (*'interviewers'*) which is equivalent to the artifact attribute *'audience _interview personname'*:

(iii) Interview date (*'date'*) which is equivalent to the artifact attribute *'datenumeric'*::

(iv) Interview duration (*'duration'*) which is equivalent to the artifact attribute *'audience _interview numeric'*:

For each business data, the plugin displays a recommended value when it is possible. Taking the example of *'Enron'*, it was recommended as an organization name value because it is the only organization name mentioned in the received email. Equally, *'Vince'* was recommended as one of the interviewers as it is mentioned in the received email as a person name. Finally, the plugin allows the user to check/uncheck each business data depending on whether he/she wants to integrate it into the next activity email or not. It additionally allows him/her to adjust the business data values. Taking the example of the interviewers business data, the administrative assistant must intervene to select more person names to be consistent with the manager request (i.e. *'Zimin Lu'*, *'Tanya Tamarchenko'* and *'Stinson Gibner'*).

We illustrate in Figure 8.6 (see the section *Business data template of the recommended activity(ies)*) the finally adopted choices for the business data templates in terms of: (i) the attributes to be included in the next activity email (i.e. organization name, interviewers and date), and (ii) the recommended/adjusted values. The figure additionally shows the generated email main body by the plugin once validating the recommended and adjusted business data template (See the *Generated Email* section in Figure 8.6). As illustrated, the

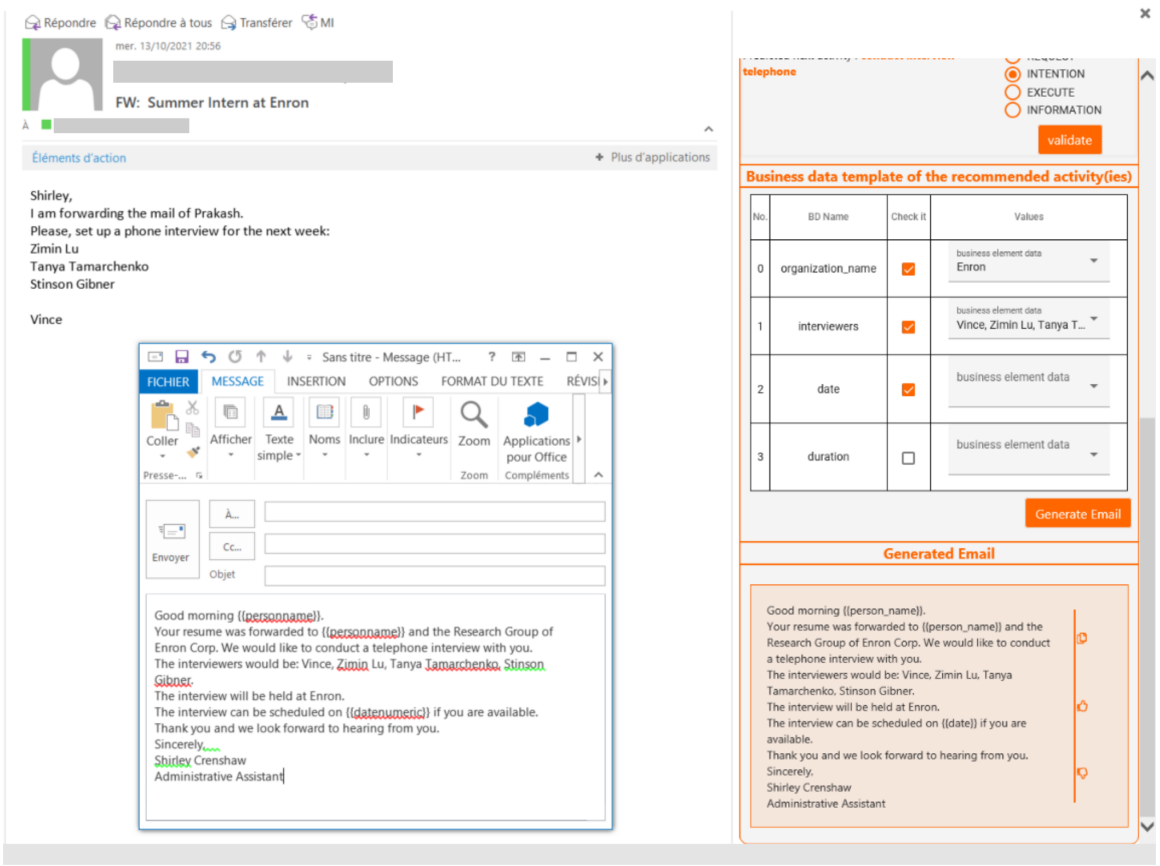
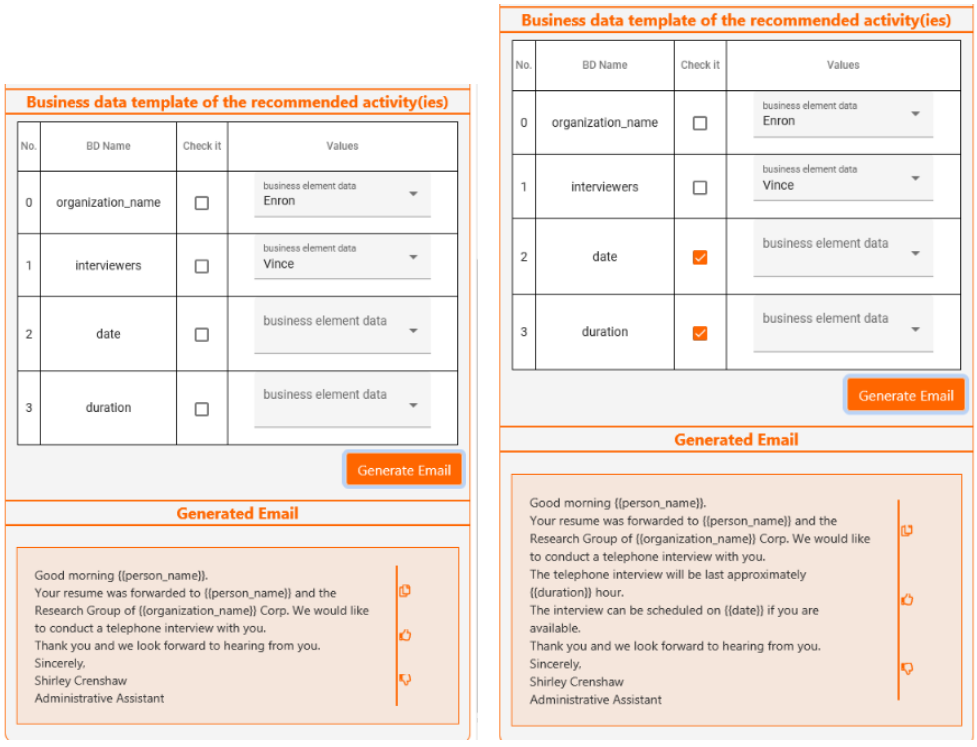


Figure 8.6: Plugin in read mode: Part 2

plugin allows user to interact with the textual recommendation functionality. For instance, he/she could give feedback on the recommended email in an explicit way (see the left/right thumbs up/down icons for expressing like/dislike). Furthermore, the user could copy the content of the recommended email to use it when writing his/her next activity email (see the copy icon in the *Generated Email* section and the new email displayed in Figure 8.6)). This kind of action is considered as an implicit action that would refer to accepting the recommendation content. Actually, we aim from these icons to collect user feedback that we judge useful for future plugin improvements. With such improvements, we target to capture events referring to user feedback for enhancing the relevance of the recommended BP elements.

As we previously explained, the generation of the recommended email depends on the selected business data attributes and the adjusted values (made by the user at the level of the section *Business data template of the recommended activity(ies)* in Figure 8.6)). For instance, the plugin fills the template sentences with the mentioned business data values. As for the non mentioned ones, it displays their location in the recommended email where they must be included by the user. Furthermore, the plugin only shows in the generated email the start and the ending sentences and the business data template sentences related to the organization name (i.e. *The interview will be held at Enron*), interviewers (i.e. *The interviewers would be personname*) and the date. The one in relation to the duration business data was not shown. We illustrate in Figure 8.7 additional examples of the generated emails after varying the selecting business data attributes. When no business data attributes is



(a) Generated email without selecting business data attributes

(b) Generated email after selecting the date and the duration business data attributes

Figure 8.7: Example of generated emails while varying the selecting business data attributes

selected (see Figure 8.7 (a)), the plugin displays only the start and the ending sentences. As when the duration and the date business data attributes are selected, the plugin includes the corresponding template sentences.

8.3 CRM data analysis for mining reasons behind users' satisfaction/ non-satisfaction

In this second, application, we extend our activity discovery solution to mine, in an unsupervised way, reasons behind users satisfaction and non-satisfaction from CRM data. More precisely, we focus on verbatim records where users express their feedback concerning some services. One of the main requirements to be full-filled is to adjust our solution to be additionally compatible with the french language (as it is the main language used by users when writing their feedback that we analyse in this section in the form of verbatim records). In what follows, we start by presenting an overview on the approach that extends our activity discovery solution for mining frequent reasons behind satisfaction/non-satisfaction (Section 8.3.1). Therefore, in Section 8.3.2, we present a use case where we apply our approach on real-life dataset to show examples of obtained results.

8.3.1 Approach Overview

As illustrated in Figure 8.8, our solution for mining reasons behind user satisfaction/non-satisfaction is composed of three main phases:

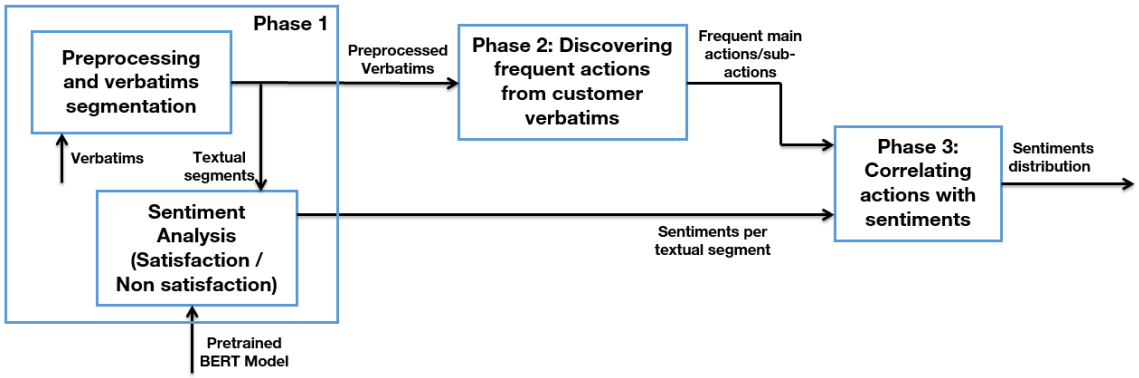


Figure 8.8: Main steps for verbatim records' analysis

Phase 1: Verbatim records preprocessing & sentiment discovery: This phase extends the same preprocessing steps described in the case of emails (Section 5.2) to be applied on verbatim records. It differs in two main points. The first point is at the level of removing stop words. In fact, it does not remove adjectives (e.g. friendly, nice, long, unexplained) as they would refer to potential important reasons for users' satisfaction/ non-satisfaction. The second point is the incorporation of a sentiment (i.e. satisfaction/non-satisfaction) discovery

component expressed by users. To this end, it relies on a pretrained BERT (Bidirectional Encoder Representations from Transformers) model for sentiment analysis [15]. This model was trained on verbatim records expressed towards amazon products to ensure a binary classification; it associates, to a given verbatim record, one label of positive or negative sentiment. In our case, we consider a positive/negative sentiment as a satisfaction/non-satisfaction sentiment.
Actually, one verbatim record would contain contradictory sentiments even in the same sentence, which would be related to different reasons (e.g. one user could say: *I am satisfied with the reception service but I am disappointed with the solution to my problem*). Therefore, multiple sentiment occurrences could be identified in one verbatim record/sentence. To deal with such situation, we adjust our step of splitting emails into sentences. We consider, in addition to punctuation, textual separators referring to contradictions (e.g. while, whereas, despite, ..) to split verbatim records into textual fragments of the smallest possible contents (in terms of complete sentence structure). Finally, we apply the sentiment discovery model on each obtained textual segment. In this way, one verbatim record will be associated with a number of sentiment occurrences, each corresponding to the one expressed in each textual segment.
Phase 2: Discover frequent actions from users' verbatim records: This phase extends the unsupervised learning solution for activity discovery to equally handle the French language. We adopt the extensions explained in Section 5.4.6 which mainly rely on: (i) the WOLF (i.e. Wordnet Libre du Français) dictionary as a dictionary of french synonyms when generating patterns of concepts, and (ii) the multi-language BERT model trained on Wikipedia data [80] for matching each pair of noun and verb having rewording relation. The ultimate goal of this phase is to discover frequent actions according to three types:

(i) Main actions: Each main action corresponds to a group of patterns sharing the same realization type, which is mainly deduced from their action verbs. It is obtained from our step grouping patterns sharing similar main actions as explained in the activity discovery part (Chapter 5, Section 5.3.3.2). The main action label is picked from the most frequent action pattern while prioritizing the one of general meaning. Taking the example of a group of action patterns including *'find solutionTvn'*, *'resolve problemTvn'* and *'resolve connection failureTvnn'*, the pattern *'find solutionTvn'* will be prioritized to represent the general meaning of the common main action between patterns;

(ii) Main action specifications/ sub-actions: They correspond to the activities discovered per each group of patterns sharing similar main action (see Section 5.3.3.4). They further specify the main action by providing additionally insights concerning: (i) its context of realization, or (ii) more detailed reasons concerning the corresponding user sentiments. Taking the example of the action *'resolve connection failure'*, it further specifies the main action *'find solution'* as it indicates the type of the problem to be/that was solved (i.e. in relation to internet connection failure);

(iii) Artifact actions: They correspond to main actions/sub-actions related to the same artifact (e.g. *'receive invoice'*, *'send invoice'* and *'check invoice'* are related to the artifact *'invoice'*);

Phase 3: Correlate actions with sentiments: This phase combines the occurrences of actions (according to their different types) with the identified sentiments of their corresponding textual segments. It enables inferring the following elements:

- Sentiment distribution around each main action;
- Actions around there are mostly satisfactions; these actions are considered as satisfaction reasons;
- Actions around there are mostly non-satisfaction; these actions are considered as non-satisfaction reasons;
- Actions/sub-actions grouped per artifact (e.g. fibre, solution, invoice, etc) & the related sentiment distribution;

8.3.2 Use case

In this part, we outline the results we obtained after applying our approach on 108,370 verbatim records of a telecom operator users. These verbatim records are related to services concerning internet and phone lines, e.g. installation, subscription, cancellation, technical and advisor support. We present in this part an overview on the discovered actions mostly inducing satisfaction/non-satisfaction. Then, we describe the visualization tool that we conceived to facilitate exploring the obtained results. Finally, we report the feedback of some business experts regarding the designed tool.

8.3.2.1 Overview on the discovered actions/Reasons

We show in Table 8.1 an extract of the most frequent main actions discovered by our approach. These main actions are organized according to two main categories: (i) Mostly inducing satisfaction (see main actions from id_1 to id_8), and (ii) mostly inducing non-satisfaction (see main actions from id_9 to id_{17}). For each main action, we provide:

- Its label (expressed in french language) as it was discovered by our approach, e.g. *'apporter_donner_fournir réponse'*;
- Its English signification, e.g. provide an answer;
- Its frequency of occurrence, e.g. 6951;
- The set of sub-actions/specifications that further explain it. Taking the example of the main action provide an answer (which mostly induces user satisfaction), we could understand from its specifications that the quality of the answer (e.g. perfect, exact, efficient, clear) and the way in which advisors give answers (e.g. attentively, lovely) are the more detailed reasons for user satisfaction. For the main action of making

Table 8.1: Frequent reasons for users' satisfaction and non-satisfaction

ID	Action/Reason (as discovered in French)	Action/Reason (English signification)	Freq	Specifications (as discovered in French)	Specifications (English signification)
Mostly inducing satisfaction					
id_1	apporter_donner _fournir réponse	provide an answer	6951	apporter réponse parfaite, efficace, claire; répondre attentivement, favorablement, charmant, correctement_exactement	provide perfect, efficient or clear answer; answer carefully, favorably, charmingly or correctly
id_2	apporter_donner explication	provide explication	1242	expliquer offre, détail, procédure, solution; expliquer agréable, compétente, correctement	explain offer, detail, procedure or solution; explain in a pleasant, competent or correctly way.
id_3	apporter_donner écoute	listen to the user	1181	écouter besoin, agréable_aimable _sympatique	listen to need; listen in pleasant or sympatic way
id_4	apporter_donner information_renseignement	provide information	803	renseigner service, demande, question; renseigner correctement, efficace	inform service, request or question; inform correctly or efficiently
id_5	adapter solution	adapt solution to user problem	319		
id_6	adapter réponse	adapt answer to user questions	291		
id_7	apporter aide	provide help	130		
id_8	comprendre besoin	understand user's need	118		
Mostly inducing non-satisfaction					
id_9	trouver solution	no solution provided	3092	résoudre problème, service, panne, ligne, situation internet, appel, intervention	solve problem, service, failure, line, internet situation, call or intervention
id_{10}	mettre attente_attente0longue	waiting	2869	attendre reponse_retour, appel, fibre_installation0fibre, facture_duplicata0facture, geste0commercial, proposition0commercial, ligne0téléphonique, rétablissement0ligne, confirmation, devis, rdv, internet, contrat, explication attendre longtemps	wait for reply, call, fiber installation, invoice, duplicate invoice, commercial gesture, commercial proposal, phone line, line restoration, confirmation, quote, appointment, internet, contract or explanation; long wait
id_{11}	aboutir_remonter demande	No answers regarding requests	1987	demander démanagement0ligne, résiliation0ligne, transfer0ligne, assurrance, rdv, devis, fibre_installation0fibre, documents, etc.	request line removal, line termination, line transfer, insurance, appointment, quote, fiber installation, documents, etc.
id_{12}	effectuer changement_modification	Difficulties in making changes or dissatisfaction with changes that were made without notifying the user	1495	changer abonnement, offre, adresse, etc	change subscription, offer, address, etc.
id_{13}	recevoir résiliation	receive cancellation	1108	annuler_résilier contrat, contrat0pro, ligne, abonnement, commande, rdv, service, demadne, offre, option, internet, intervention, installation	cancel contract, contract0pro, line, subscription, order, appointment, service, request, offer, option, internet, intervention, installation
id_{14}	couper service	stop a service	1046	couper ligne, appel, internet, adsl, fibre	cut line, call, internet, adsl, fiber
id_{15}	effectuer paiement	Problems with paiement or paying an extra-service	984	payer facture, prestation_service, prix, forfait, ligne, assurrance, téléphone, fidélité	pay bill, service, price, package, line, insurance, phone, loyalty
id_{16}	parler client	the manner of talking with cutomers	285	parler français_mal0français, parler robot	speak poorly French, speak like a robot
id_{17}	balader service	move a client from one department to another	274		

requests (i.e. id_{11} in Table 8.1) where users most often express non-satisfaction, we could understand from its specifications that these requests are related to moving/ cancelling/ transferring phone lines, fibre installation, quote, etc.

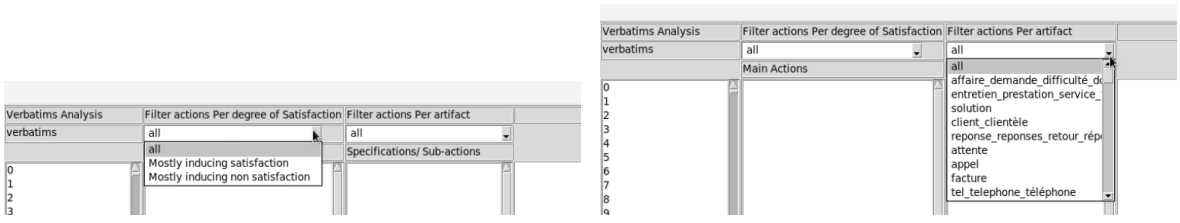
Table 8.1 shows that the reasons/actions that mostly induce a satisfaction sentiment are related to supporting users in their problems. They include: (i) user listening and understanding (i.e. id_3, id_8), (ii) providing suitable answers while adapting them and the provided solutions to their needs (id_5, id_6), and (iii) providing help, clear explications or useful information (i.e. id_2, id_4). As for the reasons that mostly induce non-satisfactions, they are related to: (i) lack of interaction towards user problems or requests (e.g.), (ii) some incidents in technical services (e.g. online payment, internet connection), (iii) making unexpected actions (e.g. cut a phone line, change some contract terms without users' consent), and (iv) the way in which advisors talk to the users (e.g. speak poorly in French or as a robot).

8.3.2.2 Visualization tool

For visualizing the obtained results, we extend our experimental tool described in Section 6.5.4. The extended tool allows users to simultaneously filter actions according to their:

(i) Category (see Figure 8.9 (a)): Three options could be selected: *'all'* to display all discovered actions, *'Mostly inducing satisfaction'* to display actions around them users most often express positive sentiments (i.e. satisfaction), and *'Mostly inducing non-satisfaction'* to display actions around them users most often express negative sentiments (i.e. non-satisfaction).

(ii) Artifact (see Figure 8.9 (b)): Discovered artifacts are displayed in the form of selection list. They are shown in decreasing order w.r.t the overall number of their action occurrences.



(a) Select action category **(b)** Select artifact

Figure 8.9: Selection lists for filtering actions

Once selecting the desired category or/and artifact options, the tool filters the discovered main actions according to the selected options. Then, it displays them in the *'Main Actions'* (i.e. block B2 in Figure 8.10) column sorted in descending order w.r.t their number of occurrences. Once picking a main action, the tool projects its specifications on the selected category and artifact. Then, it displays them in the third block *'Specifications/Sub-actions'*

(i.e. block B3 in Figure 8.10). We show in Figure 8.10 example of actions displayed when selecting the following options: *'Mostly inducing satisfaction'* as category and *'all'* as artifact (we hide a part of them in the illustrated example for confidentiality reasons). By clicking on one main action (e.g. *'apporter_donner_fournir réponse'*, i.e. provide answer), the tool displays in the third bock (B3) the set of sub-actions that further specify it, e.g. answer absolutely_completely_immediately_perfectly (*répondre absolumenet_entièrement _immédiatement _parfaitement*). In the fourth block (B4), it displays two types of visualization. The first one illustrates through a pie-chart the distribution of users' sentiments around the selected main action (e.g. 73.5% satisfaction & 26.5% non-satisfaction). The second one illustrates through a bar-diagram the distribution of these sentiments per sub-action/specification to understand the sub-actions that contribute most to the satisfaction/non-satisfaction sentiments. Finally, in the fifth block, the tool displays the verbatims where the selected main action occurred (we hide them in the illustrated example for confidentiality reasons).

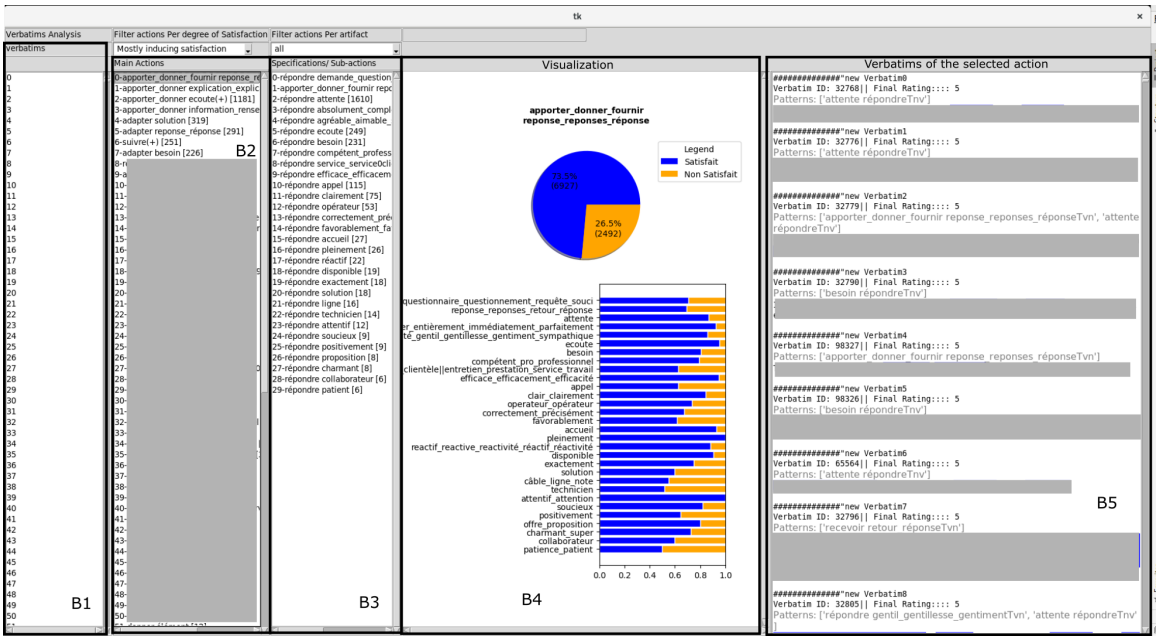


Figure 8.10: Experimental tool for verbatim record analysis: Display a satisfaction main action

Actually, organisms generally target enhancing the quality of their services. That is why they most often seek to know the reasons of users' non- satisfaction even if they occur at low frequency. This allows them to identify services requiring further improvements. We show in what follows two examples of functionalities allowing to satisfy this need through our tool.
In the first functionality example, we show in Figure 8.11 a set of actions displayed when selecting the following options: *'Mostly inducing non-satisfaction'* as category and *'all'* as artifact. By clicking on the main action (e.g. *'mettre attente'*, i.e. wait), the tool displays in the third bock (B3) the set of sub-actions that further specify it, e.g. wait answer (*attendre réponse*), wait call (*attendre appel*), etc. We understand from the bar-diagram in the fourth block (B4) that waiting for a long time is the sub-action that mostly contributes in the non-satisfaction sentiment, followed by waiting an answer.

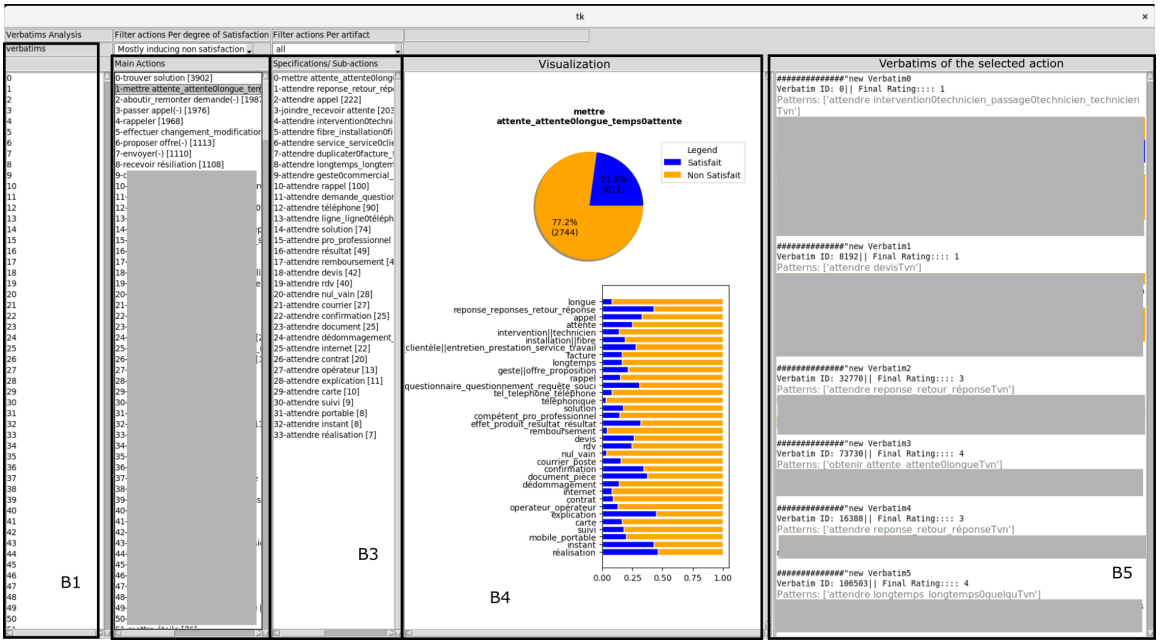


Figure 8.11: Experimental tool for verbatim record analysis: Display a non-satisfaction main action

In the second functionality example, we pick one artifact (i.e. *'fibre'*) from the list displayed in Figure 8.9 (b). The tool displays in the second block (B2) the main actions that act on it and that are filtered according to the selected category. Figure 8.12 shows the results that

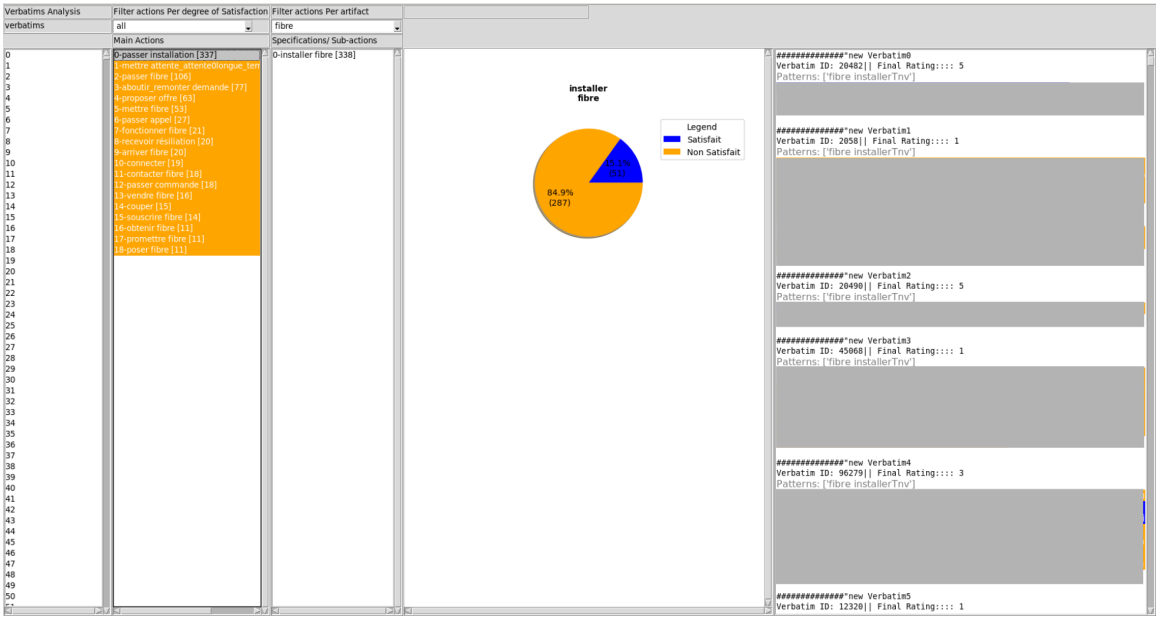


Figure 8.12: Users' satisfaction/non-satisfaction with fiber service

we obtained (by additionally selecting the category *'all'*). Main actions that are most often expressed with negative (positive) sentiments are colored in orange (blue). We could notice that all main actions frequently expressed in the context of the artifact *'fibre'* are colored in orange, which means that all of them induce users' non-satisfaction. Taking the example of

the action *'installer fibre'* (i.e. install fibre), the corresponded pie-chart shows that around 85% of the expressed sentiments are negative. By checking the related verbatim records, we could understand that this due to: (i) technical problems that are most often faced, (ii) lack of technical expertise in the face of some problems.

8.3.2.3 Business experts feedback

To assess the usefulness of our results, we have consulted the business experts who are specialized in user relations and who carried out manual analysis on the same dataset of verbatim records. These experts have drawn the following feedback:

- The tool facilitates and speeds up the process of exploring a big set of verbatim records.
- The obtained results bring out additional insights on the actions/reasons that induce users' satisfaction/non-satisfaction, and that have not been noticed previously. We cite the way the advisor talks to the users.
- In practice, we could view such results useful for conceiving a warning system bringing out alerts concerning: (i) services of poor quality deduced from the artifacts that are highly correlated with non-satisfaction sentiments, (ii) the corresponding reasons which are deduced from the related discovered actions. Each alert is useful for becoming aware of execution problems related to a main action or an artifact. The goal is to invite user service managers to study in depth the reasons. They could for example check/read the related verbatim records or establish further surveys if the discovered specifications/sub-actions do not provide sufficient explications. The ultimate goal is to capture the real reasons for proposing the suitable solutions for these execution problems in order to enhance service quality;
- The implemented tool could be further extended to ensure a daily analysis of users' verbatim records. In this way, alerts concerning one artifact/service could be monitored to check if they decrease or not over time. This is useful for judging the effectiveness of the proposed solutions for overcoming execution problems.
- Manual intervention could be allowed, namely for business experts, to define word synonyms and the degree of granularity of artifacts.

8.4 Conclusion

In this chapter, we presented potential applications that extend the algorithms that we introduced for BP discovery from emails. We focused on one potential application where we have adapted our solution for discovering activities to enable mining the reasons behind user satisfaction and non-satisfaction. Furthermore, we explained, in a second potential application, how our overall results could be extended to a BP discovery & recommendation tool to

be integrated in emailing systems. We additionally presented a first version for concretely implementing the recommendation component of such tool. For both of these applications, we were based on example of results/scenario of use to show their business utility.

In summary, we showed through this chapter how our introduced algorithms would be applicable in other contexts (i.e. analysis of users' verbatim records), and how they open the door to *smart* functions in emailing systems and BPM. In the next and the final chapter, we further discuss potential perspectives and improvements of these applications and of the overall work presented in this report.

Chapter 9

Conclusion & Perspectives

In this chapter, we first summarize our contributions in this thesis to discover BP from emails. Then, we discuss our future research directions.

9.1 Contributions

Information systems (IS) are nowadays more and more integrated in the companies. They aim to manage the underlying BP from one hand, and to enhance the efficiency of the actors involved in BP from another hand. Therefore, the mass of stored data about BP executions is becoming increasingly important. BP discovery techniques have appeared to turn such data into valuable knowledge giving insights on how BP are actually executed. These techniques are mostly limited to analyzing structured execution logs of IS that support partially or totally BP. Traces of BP that are executed outside such IS are most often ignored. Particularly, emailing systems are widely involved as an alternative or as a complement to manage business activities. Email logs could then report additional BP execution traces, making them a valuable data source for BP discovery. Given the non structured nature of email logs data, traditional BP discovery algorithms could not be applied (or at least not directly applied) on emails. More important, the way emails are used for BP execution is not formalized in the literature.

In this thesis, we worked on two main objectives in order to accommodate BP discovery from emails. We aimed specifically to: (1) Define the BP knowledge (i.e. BP perspectives and BP elements) that could be discovered from emails, and (2) Automate the discovery of the defined BP knowledge from emails: (i) without disposing a priori knowledge about them, and (ii) while allowing the discovery of multiple BP elements per one email.

For the first objective, we defined, inline with the specifities of email usage in BP context, four BP fragment perspectives: functional, data, actors and behavioral perspectives. We formalized the definition of these perspectives and their relation with the unstructured log data of emails. We described such relation through a metamodel that outlines the main entities ensuring email log transformation into a structured event log and then into a valuable BP knowledge.

For the second objective, we introduced an approach that: (i) is totally unsupervised, and (ii) does not require a prior knowledge about BP or exhaustive human intervention. The proposed approach automate the generation of a structured event log from an email log data.

Then, it automates the discovery of multiple BP perspectives from the generated event log. To convert an email log into a structured event log, we introduced a pattern discovery based approach to discover frequent activities while allowing the identification of multiple ones per email. We defined each activity as the composition of an activity name and a set of business information. We considered that multiple actors could intervene with different contributions not limited to performing it. We discovered each activity component in the form of a set of patterns reflecting the combinations of words frequently used by employees to express it. Therefore, we used the obtained activity characterization in terms of patterns to discover their occurrences in emails. Afterwards, we identified their related speech acts to recognize senders' purposes when including them in emails.
To discover BP fragments w.r.t the defined perspectives, we introduced several algorithmic solutions for mining the generated event log: (i) overlapping clustering of activities to discover their manipulated artifacts on the basis of their business information similarity, (ii) overlapping clustering of BP elements (i.e. activities, artifacts and activity actors) to discover BP fragments, (iii) discovering activity actor perspective from activity occurrences and their related speech acts, and (iv) mining sequencing constraints between event types to discover the behavioral perspective.

To validate our approach, we were based on emails retrieved from the public Enron dataset. We carried out different experiments covering the various phases, parts and steps in our overall framework. We additionally shared the obtained results (in terms of the discovered BP knowledge) to allow quantitative comparisons with future studies if they use the same public data set, which is absent in existing approaches (link [1]). Finally, we show the potential of our proposals to be applied on other datasets such as those of Orange emails and verbatim records of a telecom operator .

The design principles we presented in the introduction (Section 1.3.1) have been respected:

- Consistency (Principle 1): We defined the BP knowledge to be discovered from emails in accordance with the specifities of emailing systems. We integrated the notion of sender purposes when including activities in emails to define: (i) new event types, and (ii) actor contributions to characterize how they intervene in the activities . We additionally adopted a declarative approach in the discovery of the BP behavioral perspective to take into account with the poorly structured nature of the BP fragment performed through emails.

- Automation (Principle 2): We proposed an automated approach for BP discovery from emails that integrates automated techniques such as frequent pattern discovery from emails, clustering patterns into activities, generating a structured event log from emails, discovering artifacts and BP fragments and mining the generated event log for discovering BP fragments w.r.t multiple perspectives;

- Data robustness (Principle 3): We showed this through the different experiments that we have conducted. This concerns basic parts in our approach, namely the unsupervised

[1] http://www-inf.it-sudparis.eu/SIMBAD/tools/processDiscoveryFromEmails/

learning part for activity discovery, which could be applied on various types of data (i.e. emails, verbatim records) written: (i) using more than one language (i.e. English and French), (ii) by different actors (i.e. employees, service users), and (iii) belonging to more than one organism (e.g. Enron, Orange, Telecom operator);

- Balanced Recall (Principle 4): During the evaluation, we assessed the performances of the activity discovery approach in terms of recall and degree of relevance of the discovered activities/patterns. We picked parameter values that make a comprise between them;

Our work have the following main advantages. The first one is to propose a completely unsupervised approach, which facilitates knowledge extraction without human intervention. This is useful for practical applications due to two main reasons:

- BP activities differ from one employee to another even in the same organism. Therefore, they could not be defined in advance;
- Emails are considered as private and confidential data. Therefore, real emails of employees could not be available in advance to be annotated (in the case of supervising learning approaches). Additionally, asking employees to do such task locally (i.e. in their mail boxes) is not feasible as they rarely accept to be involved in time consuming tasks.

The second advantage is related to the discovery of new perspectives (i.e. data and organizational). Various business data types (e.g. price, quantity) used during activities' execution were discovered rather than predefined ones (e.g. documents, link). A new definition for organizational perspective was introduced while considering that, for performing one activity, multiple actors with different contribution types could intervene. Finally, the third advantage is that a public dataset (i.e. Enron) was used for evaluation purposes while sharing the obtained results. This allows quantitative comparisons with future studies if they use the same public data set, which is absent in existing approaches.

Last but not least, our approach could be extended to provide an automated support for other approaches. We showed this through the two potential applications of: (i) verbatim records' analysis for discovering reasons behind users' satisfaction/ non-satisfaction, and (ii) recommendation system of BP knowledge integrated in email clients.

9.2 Perspectives

In the future work, we intend to: (i) Consider various data sources for BP discovery and recommendation purposes (Section 9.2.1), (ii) Improving the recommendation system (Section 9.2.2) (iii) Meet the practical integration requirements for an email analysis tool in real life (Section 9.2.3)

9.2.1 Consider various data sources for BP discovery and recommendation purposes

This perspective includes two potential axis. In the first axis, additional data sources in relation to the execution logs of IS that totally or partially support BP (e.g. BPMS), could be integrated. On the one hand, these execution logs could assist the steps of analysing emails to enhance the relevance and the recall of the extracted BP knowledge. On the other hand, they could provide additional relevant BP knowledge that were not necessarily mentioned in emails. This would provide a more complete view of the BP execution. In the second axis, email user feedback could be integrated as a data source that would continuously improve the discovered and the recommended BP knowledge. To this end, additional functionalities in the BP discovery & recommendation tool have to be integrated to capture the event log of user implicit/explicit reactions towards the provided results. The goal is to enable the analysis of such feedback to improve the relevance of the discovered and the recommended BP knowledge for future emails, e.g. by learning the set of values that are most often preferred by the user for a given business data.

9.2.2 Improving the recommendation system

To further automate the recommendation system and improve its efficiency, additional research questions have to be further studied:

(1) How to identify the most related BP elements to the email on read or compose mode (i.e. received email or email being written)? For instance, further recommendation algorithms have to be investigated to bring out the most relevant business data and activities according to the email BP context and interlocutors;

(2) How to generate and recommend a natural language text in relation to the identified BP elements? More natural language generation techniques could be studied to automate the construction of the recommended emails. This is valid for the auto-completion techniques in order to support workers when composing their emails. The goal of these techniques is to suggest useful expressions or business data values related to the activities being typed into an email;

9.2.3 Meet the practical integration requirements

Due to the confidentiality issue of emails, applications analysing their textual contents generally encounter several constraints when aiming integrating them in real life. In future studies, we target to work on two main axis to further meet the practical integration requirements of a BP discovery & recommendation system in emailing systems:

(1) **Meet the confidentiality & data privacy constraints**: This requires investigating additional research questions: (1) How to store the discovered BP knowledge at the

level of each email client and according to which storage structure ? and (2) How to securely ensure the sharing of the validated ones to a central sub-system ? For instance, we presented a vision for not disclosing the real contents of relevant and confidential information values while allowing sharing an anonymized version of them at the level of each employee mail box. This vision is based on integrating a hashing process to be applied on these information. As we previously mentioned, the hashing process is irreversible. However, it would be limited from confidentiality perspective if we simultaneously know: (i) the used algorithm when applying it, and (ii) the real value (e.g. an email address of a user) on which we want to look for. In fact, we could obtain the corresponding hash code and use the stored data at the level of the central sub-system to disclose related details (e.g. activities). One of the possible solutions to be investigated is to study the *salting* concept for adding a layer of security to the hashing process. The basic idea is to add a salt (i.e. additional value) at the end of the relevant information to be hashed, which would complicate its cracking process;

(2) **Conduct study for finding the best strategy for users to accept an email analysis tool:** Email users would be afraid that an email analysis tool discloses their sensitive data (e.g. by mistake at the level of user validation or an algorithmic step). Additionally, they generally do not accept to waste a lot of time to ensure an important step in the operation of the tool. To concretely handle such constraints, a study could be carried out to find the best strategy for users to accept an email analysis tool. This study would cover two main aspects. The first one is a sociology aspect where the users' reasons for not liking to install a tool on their mailboxes could be further investigated. In the second aspect, the efficiency degree of the tool on the basis on quantified metrics could be studied.

Appendix A

Résumé Etendu

A.1 Contexte et problématique de la recherche

La fouille de processus vise à analyser les traces d'exécution des systèmes d'information (SI), utilisés dans le cadre des activités métiers, pour découvrir des connaissances sur les processus métiers (PM). La découverte de processus consiste généralement à identifier des modèles de processus métier (PM) réels à partir de journaux d'événements structurés. D'importants travaux de recherche ont été menés dans ce domaine. Cependant, ils supposent généralement que ces traces d'exécution ont un niveau de structuration élevé. Cela signifie que: (i) ils sont composés d'enregistrements structurés, chacun capturant l'exécution d'une activité, et (ii) une partie des attributs des événements d'exécution (comme le nom de l'activité, l'horodatage) sont explicitement inclus dans ces enregistrements, ce qui facilite leur inférence. Néanmoins, les PM peuvent être entièrement ou partiellement réalisés dans des SI moins structurés générant des traces d'exécution de faible niveau de structuration. Les systèmes de courriels sont largement utilisés pour réaliser de manière collaborative des activités de PM. Compte tenu de la propagation des environnements de travail numériques, les systèmes de courriels sont de plus en plus utilisés, ce qui fait de leurs traces d'exécution une source de données importante pour la découverte des PM.

Du point de vue de la gestion des PM, l'exécution d'activités par le biais de courriels facilite les interactions entre les différents acteurs des PM et favorise le partage de l'expertise métier. Cependant, cela induit également certains inconvénients, notamment en l'absence d'outils permettant la gestion des traces d'exécution correspondantes. D'une part, ces traces d'exécution sont susceptibles de ne pas être gérées en raison du format non exploitable de leur stockage et de leur faible niveau de structuration. Elles pourraient être incluses dans les traces des échanges des courriels avec une incertitude concernant la présence des éléments de PM qui y sont liés (par exemple, le nom de l'activité, le nom du PM, les données métiers manipulées, l'instance du PM). Même dans le cas de leur présence, ils sont exprimés en langage naturel. Cela signifie qu'ils ne sont pas explicitement séparés ou identifiés pour être stockés selon une structure spécifique. D'autre part, une intervention manuelle reste nécessaire lors de la réalisation des activités de PM par le biais de courriers électroniques. Si elles sont effectuées de manière répétée, elles risquent de prendre beaucoup de temps, ce qui conduit à la nécessité d'automatiser leur exécution.

La découverte de PM à partir des courriels électroniques est donc importante pour assurer une meilleure gestion des PM. Cependant, en raison de la nature non structurée des contenus

textuels des courriels, les techniques les plus courantes de découverte de processus ne peuvent pas être directement appliquées. Les approches existantes qui découvrent les PM à partir de courriels sont généralement supervisées ou nécessitent une intervention humaine importante. Elles se sont concentrées sur la découverte de PM en fonction de leur perspective fonctionnelle (qui définit l'objectif métier à réaliser à la fin de l'exécution d'un PM ou l'un de ses activités élémentaires) et comportementale (qui définit les conditions d'exécution des activités) tout en négligeant la découverte de leur perspective de données (qui définit les entités informationnelles manipulées par les activités de PM) et de leur perspective organisationnelle (qui décrit les acteurs impliqués dans les activités de PM). En outre, ils n'ont pas étudié comment les systèmes de courriels sont utilisés dans le contexte des exécutions des PM. Ils supposent que les systèmes de courriels sont utilisés de la même manière que les employés utilisent les SI ordinaires supportant l'exécution des activités de PM. Cependant, les employés utilisent les courriels pour exécuter des fragments de PM peu structurés (c'est-à-dire des parties) plutôt que des PM complets et bien structurés. Ces fragments de PM ne sont pas nécessairement connus à l'avance, ce qui induit la nécessité de découvrir la perspective fonctionnelle des PM. En outre, les employés utilisent des courriels à des fins différentes lorsqu'ils parlent des activités des PM (par exemple, des informations sur l'exécution, la demande ou la planification de l'exécution des activités, etc.). Il en résulte l'apparition de nouveaux types d'événements faisant référence à l'objectif de l'expression des activités dans les courriels plutôt que des événements se référant uniquement à leur exécution.

A.2 Objectifs et contributions

Dans cette thèse, nous avons pour objectif de découvrir des fragments de PM en analysant les courriels électroniques. L'étude est menée au sein du groupe Orange et fait partie du projet "Orange Process Discovery". Nos principaux objectifs sont les suivants:

- **Objectif 1:** Définir les connaissances PM (c'est-à-dire les perspectives et les éléments de PM) qui pourraient être découverts à partir des courriels électroniques;
- **Objectif 2:** Automatiser la découverte des connaissances PM définies à partir des courriels: (i) sans disposer de connaissances a priori à leur sujet, et (ii) en permettant la découverte de plusieurs éléments de PM par courriel. Cet objectif nécessite la réalisation de ces deux sous-objectifs:
  - **Objectif 2.1:** Automatiser la génération d'un journal d'événements structuré à partir des courriels éléctroniques;
  - **Objectif 2.2:** Automatiser la découverte des perspectives de PM à partir du journal d'événements généré;

Pour le premier objectif, nous définissons quatre perspectives de PM à découvrir à partir des courriels électroniques: Fonctionnelle, organisationnelle, données et comportementale. Pour la perspective fonctionnelle, nous adoptons la notion de "fragment de PM" (c'est-à-dire

un ensemble d'activités définissant une partie d'un PM) à découvrir à partir des courriels plutôt qu'un PM complet en raison de l'incomplétude des traces des PM dans les courriels. En ce qui concerne la perspective organisationnelle, nous la décrivons comme l'ensemble des acteurs et des groupes d'acteurs impliqués dans l'exécution de chaque activité, leurs contributions connexes et la façon dont ils interagissent globalement les uns avec les autres afin d'exécuter chaque activité. Nous intégrons la notion de « acte de parole » de l'expéditeur qui définit son objectif de l'expression des activités métiers dans ses courriels. Nous utilisons cette notion pour définir les contributions des acteurs et de caractériser leurs interactions. Pour la perspective des données, nous la définissons comme l'ensemble des artefacts (c'est-à-dire des entités informationnelles) manipulés par les activités du fragment de PM et les relations entre eux. Quant à la perspective comportementale, nous la décrivons par un ensemble de contraintes séquentielles entre trois types d'événements. Ces types d'événements définissent de quelle perspective nous pouvons voir l'occurrence d'une activité dans un courriel, ce qui correspond au : (i) nom de l'activité survenue, (ii) l'objectif connexe de l'expéditeur lorsqu'il l'inclut dans le courriel, ou (iii) à la combinaison entre (i) et (ii) pour mieux refléter la façon dont les courriels sont utilisés dans le contexte des PM.

Nous proposons un méta-modèle pour formaliser la relation entre les données d'un journal de courrier électronique et nos perspectives définies [28]. Nous introduisons une structure de journal des événements compatible avec la découverte des perspectives définies. Nous définissons également les éléments de PM assurant la transformation des traces d'echanges des courriels en cette structure de journal d'évènements.

En ce qui concerne le deuxième objectif, nous introduisons une approche qui : (i) est totalement non-supervisée, et (ii) ne nécessite pas de connaissances préalables sur les PM ou une intervention humaine exhaustive. Notre approche transforme une trace d'échange des courriels électoniques en un journal d'événements structuré, puis l'exploite pour découvrir de multiples perspectives de PM.

Pour générer un journal d'événements structuré à partir des traces d'échanges des courriels éléctroniques, nous introduisons une approche basée sur la découverte de patrons de mots [29] pour découvrir les activités fréquentes tout en permettant la détection de plusieurs activités par email. Nous définissons chaque activité à ce niveau comme la composition d'un nom d'activité et d'un ensemble de données métiers manipulées. Nous découvrons chaque composante de l'activité sous la forme d'un ensemble de patrons reflétant les combinaisons de mots fréquemment utilisées par les employés pour l'exprimer dans les courriels. Par conséquent, nous utilisons la caractérisation des activités obtenue en termes de patrons pour découvrir leurs occurrences dans les courriels. Nous proposons ensuite d'identifier les actes de paroles qui leur sont liés afin de reconnaître les intentions des expéditeurs lorsqu'ils les incluent dans les courriels.

Pour découvrir les fragments de PM en fonction des perspectives définies, nous introduisons plusieurs solutions algorithmiques pour : (i) le regroupement par chevauchement des activités pour découvrir leurs artefacts manipulés sur la base de la similarité de leurs informations commerciales, (ii) le regroupement par chevauchement des éléments de PM (c'est-à-dire les activités, les artefacts et les acteurs de l'activité) pour découvrir les fragments de PM,

(iii) la découverte de la perspective organisationnelle de l'activité à partir des occurrences de l'activité et de leurs actes de parole connexes [31], et (iv) l'exploitation des contraintes séquentielles entre les types d'événements déduits des activités et des actes de parole pour découvrir la perspective comportementale.

Toutes nos approches ont été mises en œuvre et évaluées à l'aide d'un ensemble de données de courrier électronique public Enron. Nous avons rendu publics nos résultats expérimentaux (see this link[1]) afin d'assurer la faisabilité d'un type de comparaison avec les travaux relatifs, ce qui permet une analyse plus pratique pour les recherches futures. Nous montrons enfin l'utilité de nos résultats pour améliorer la gestion des PM à travers deux applications: (i) un outil de découverte et de recommandation des connaissances de PM à intégrer dans un système de gestion de courriels, et (ii) l'analyse de données CRM pour l'exploration des raisons de la satisfaction/non-satisfaction des utilisateurs.

[1] http://www-inf.it-sudparis.eu/SIMBAD/tools/processDiscoveryFromEmails/

Bibliography

[1] Wil Van der Aalst, Ton Weijters, and Laura Maruster. "Workflow mining: Discovering process models from event logs". In: *IEEE transactions on knowledge and data engineering* 16.9 (2004), pp. 1128–1142 (cit. on pp. 3, 10, 21, 148).

[2] Wil M. P. van der Aalst. "Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data". In: *SEFM* 11724 (2019), pp. 3–25 (cit. on pp. 21, 51).

[3] Wil MP Van der Aalst, AK Alves De Medeiros, and Anton JMM Weijters. "Genetic process mining". In: *International conference on application and theory of petri nets*. Springer. 2005, pp. 48–69 (cit. on pp. 3, 10, 21, 148).

[4] Wil MP van der Aalst and Andriy Nikolov. "EMailAnalyzer: an e-mail mining plug-in for the ProM framework". In: *BPM Center Report* (2007) (cit. on pp. 10, 30, 35–37).

[5] Will van de Aalst. "Process discovery: Capturing the invisible". In: *IEEE Computational Intelligence Magazine* 5.1 (2010), pp. 28–41 (cit. on p. 3).

[6] Rakesh Agrawal, Ramakrishnan Srikant, et al. "Fast algorithms for mining association rules". In: *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. Citeseer. 1994, pp. 487–499 (cit. on p. 84).

[7] J. Chambers Alexander et al. "Automated Business Process Discovery from Unstructured Natural-Language Documents". In: *AI4BPM* (2020) (cit. on pp. 32, 35).

[8] Tony Allard et al. "A dataset to facilitate automated workflow analysis". In: *PloS one* 14.2 (2019), e0211486 (cit. on p. 34).

[9] S Baadel, F Thabtah, and J Lu. "Multi-cluster overlapping k-means extension algorithm". In: *proceedings of the XIII International Conference on Machine Learning and Computing, ICMLC'2015* (2015) (cit. on p. 149).

[10] Said Baadel, Fadi Thabtah, and Joan Lu. "Overlapping clustering: A review". In: *2016 SAI Computing Conference (SAI)* (2016), pp. 233–237 (cit. on p. 149).

[11] Rolf Banziger, Artie Basukoski, and Thierry Chaussalet. "Discovering Business Processes in CRM Systems by leveraging unstructured text data". In: *20th International Conference on High Performance Computing and Communications* (2018), pp. 1571–1577 (cit. on pp. 6, 30, 35, 37, 147).

[12] Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. "Fast optimal leaf ordering for hierarchical clustering". In: *Bioinformatics* 17.suppl_1 (2001), S22–S29 (cit. on p. 96).

[13] Dina Bayomie et al. "Deducing case IDs for unlabeled event logs". In: *International Conference on Business Process Management*. Springer. 2016, pp. 242–254 (cit. on p. 19).

[14] Bettina Berendt et al. "The impact of site structure and user environment on session reconstruction in web usage analysis". In: *International workshop on mining web data for discovering usage patterns and profiles*. Springer. 2002, pp. 159–179 (cit. on pp. 2, 20).

[15] Théophile Blard. *Théophile Blard, French sentiment analysis with BERT, GitHub repository*. 2020 (cit. on p. 204).

[16] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. "Density-based clustering based on hierarchical density estimates". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, pp. 160–172 (cit. on p. 96).

[17] Yen-Liang Chen and Hui-Ling Hu. "An overlapping cluster algorithm to provide non-exhaustive clustering". In: *European Journal of Operational Research* 173.3 (2006), pp. 762–780 (cit. on p. 149).

[18] William W Cohen, Vitor R Carvalho, and Tom M Mitchell. "Learning to classify email into "speech acts"". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2004), pp. 309–316 (cit. on pp. 6, 10, 24, 27, 28, 35–37, 78).

[19] Linda M Collins and Clyde W Dent. "Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions". In: *Multivariate Behavioral Research* 23.2 (1988), pp. 231–242 (cit. on pp. 104, 171).

[20] Dusanka Dakic et al. "BUSINESS PROCESS MINING APPLICATION: A LITERATURE REVIEW." In: *Annals of DAAAM & Proceedings* 29 (2018) (cit. on p. 21).

[21] Jochen De Weerdt et al. "Active trace clustering for improved process discovery". In: *IEEE Transactions on Knowledge and Data Engineering* 25.12 (2013), pp. 2708–2720 (cit. on p. 3).

[22] Claudia Di Ciccio and Massimo Mecella. "MINERful, a mining algorithm for declarative process constraints in MailOfMine". In: *Department of Computer and System Sciences Antonio Ruberti Technical Reports* 4.3 (2012) (cit. on pp. 6, 30, 33, 35, 37, 147).

[23] Claudio Di Ciccio et al. "Declarative Process Discovery with MINERful in ProM." In: *BPM (Demos)*. 2015, pp. 60–64 (cit. on p. 21).

[24] Claudio Di Ciccio et al. "MailOfMine–analyzing mail messages for mining artful collaborative processes". In: *International Symposium on Data-Driven Process Discovery and Analysis* (2011), pp. 55–81 (cit. on pp. 6, 30, 33, 35–37, 147, 148).

[25] Mark Dredze, Tessa Lau, and Nicholas Kushmerick. "Automatically classifying emails into activities". In: *Proceedings of the 11th international conference on Intelligent user interfaces* (2006), pp. 70–77 (cit. on pp. 6, 24, 25, 35, 37, 192).

[26] Marlon Dumas et al. *Fundamentals of business process management*. Vol. 1. Springer, 2013 (cit. on p. 2).

[27] Schahram Dustdar and Robert Gombotz. "Discovering web service workflows using web services interaction mining". In: *International Journal of Business Process Integration and Management* 1.4 (2006), pp. 256–266 (cit. on pp. 2, 20).

[28] Marwa Elleuch et al. "A Meta Model for Mining Processes from Email Data". In: *2020 IEEE International Conference on Services Computing (SCC)*. IEEE. 2020, pp. 152–161 (cit. on pp. 52, 221).

[29] Marwa Elleuch et al. “Discovering Activities from Emails Based on Pattern Discovery Approach”. In: *Business Process Management Forum - BPM Forum 2020, Seville, Spain, September 13-18, 2020, Proceedings*. Lecture Notes in Business Information Processing 392 (2020). Ed. by Dirk Fahland et al., pp. 88–104 (cit. on pp. 25, 27, 79, 221).

[30] Marwa Elleuch et al. “Discovering Business Processes And Activities From Messaging Systems: State-Of-The Art”. In: *29th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2020, Virtual Event, France, September 10-13, 2020*. IEEE, 2020, pp. 137–142 (cit. on p. 18).

[31] Marwa Elleuch et al. “Discovery of Activities’ Actor Perspective from Emails based on Speech Acts Detection”. In: *2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020* (2020). Ed. by Boudewijn F. van Dongen, Marco Montali, and Moe Thandar Wynn, pp. 73–80 (cit. on pp. 25, 27, 123, 222).

[32] Marwa Elleuch et al. “Multi-perspective business process discovery from messaging systems: State-of-the art”. In: *Concurrency and Computation: Practice and Experience* (2021), e6642 (cit. on p. 18).

[33] Marwa Elleuch et al. “Process Fragments Discovery from Emails: Functional, Data and Behavioral Perspectives Discovery”. In: *Information Systems* INFOSYS-D-21-00189 (2021) (cit. on p. 150).

[34] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *kdd*. Vol. 96. 34. 1996, pp. 226–231 (cit. on p. 96).

[35] Anahita Farhang Ghahfarokhi et al. “OCEL: A Standard for Object-Centric Event Logs”. In: *European Conference on Advances in Databases and Information Systems*. Springer. 2021, pp. 169–175 (cit. on p. 21).

[36] Christian W Günther and Anne Rozinat. “Disco: Discover Your Processes.” In: *BPM (Demos)* 940 (2012), pp. 40–44 (cit. on p. 21).

[37] Christian W Günther and Wil MP Van Der Aalst. “Fuzzy mining–adaptive process simplification based on multi-perspective metrics”. In: *International conference on business process management*. Springer. 2007, pp. 328–343 (cit. on pp. 3, 10, 21, 148).

[38] Jiawei Han, Jian Pei, and Yiwen Yin. “Mining frequent patterns without candidate generation”. In: *ACM sigmod record* 29.2 (2000), pp. 1–12 (cit. on p. 84).

[39] Jiawei Han et al. “Frequent pattern mining: current status and future directions”. In: *Data mining and knowledge discovery* 15.1 (2007), pp. 55–86 (cit. on p. 84).

[40] John A Hartigan and Manchek A Wong. “Algorithm AS 136: A k-means clustering algorithm”. In: *Journal of the royal statistical society. series c (applied statistics)* 28.1 (1979), pp. 100–108 (cit. on p. 96).

[41] Matthew Honnibal and Mark Johnson. “An improved non-monotonic transition system for dependency parsing”. In: *EMNLP* (2015), pp. 1373–1378 (cit. on p. 126).

[42] Frank Höppner et al. *Fuzzy cluster analysis: methods for classification, data analysis and image recognition*. John Wiley & Sons, 1999 (cit. on p. 149).

[43] Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. "Semi-supervised speech act recognition in emails and forums". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing* 3 (2009), pp. 1250–1259 (cit. on pp. 6, 10, 25, 27, 28, 35, 37, 78).

[44] Diana Jlailaty, Daniela Grigori, and Khalid Belhajjame. "A framework for mining process models from emails logs". In: *arXiv preprint* (2016) (cit. on pp. 10, 30).

[45] Diana Jlailaty, Daniela Grigori, and Khalid Belhajjame. "Business process instances discovery from email logs". In: *SCC* (2017), pp. 19–26 (cit. on pp. 31, 32, 35).

[46] Diana Jlailaty, Daniela Grigori, and Khalid Belhajjame. "Mining business process activities from email logs". In: *ICCC* (2017), pp. 112–119 (cit. on pp. 6, 10, 30–32, 35–37, 147).

[47] Diana Jlailaty, Daniela Grigori, and Khalid Belhajjame. "Mining Business Process Information from Email Logs for Business Process Models Discovery". In: (2019) (cit. on pp. 10, 31, 35, 36).

[48] Diana Jlailaty, Daniela Grigori, and Khalid Belhajjame. "On the elicitation and annotation of business activities based on emails". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (2019), pp. 101–103 (cit. on pp. 6, 10, 25, 27–29, 35–37, 78, 148).

[49] Rinat Khoussainov and Nicholas Kushmerick. "Email Task Management: An Iterative Relational Learning Approach." In: *CEAS* (2005) (cit. on pp. 6, 24–26, 35, 37).

[50] Rinat Khoussainov and Nicholas Kushmerick. "Relational learning for email task management". In: *International Joint Conferrence on ARTIFICIAL INTELLIGENCE* 19 (2005), p. 1610 (cit. on pp. 6, 24, 25, 35, 37).

[51] Nicholas Kushmerick and Tessa Lau. "Automated email activity management: an unsupervised learning approach". In: *Proceedings of the 10th international conference on Intelligent user interfaces* (2005), pp. 67–74 (cit. on pp. 26, 30, 31, 35, 37).

[52] Nicholas Kushmerick et al. "Activity-centric email: A machine learning approach". In: *Proceedings of the National Conference on Artificial Intelligence* 21.2 (2006), p. 1634 (cit. on pp. 6, 26, 30, 37, 78, 147).

[53] Michal Laclavík et al. "Tools for email based recommendation in enterprise". In: *International Conference on ENTERprise Information Systems*. Springer. 2010, pp. 209–218 (cit. on p. 197).

[54] Nassim Laga, Mohammed Oussama Kherbouche, and Pierre-Aymeric Masse. "Communication-based business process task detection-application in the crm context". In: *IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)* (2016), pp. 1–8 (cit. on pp. 6, 24, 27, 29, 35).

[55] Nassim Laga et al. "Emails Analysis for Business Process Discovery". In: *ATAED* (2019), p. 54 (cit. on pp. 6, 10, 30, 32, 33, 35–37, 78, 147).

[56] Guangming Li et al. "Extracting object-centric event logs to support process mining on databases". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2018, pp. 182–199 (cit. on p. 3).

[57] Guangming Li et al. “Extracting Object-Centric Event Logs to Support Process Mining on Databases”. In: *Information Systems in the Big Data Era - CAiSE Forum 2018, Proceedings*. Vol. 317. 2018, pp. 182–199 (cit. on p. 21).

[58] Xixi Lu et al. “Discovering interacting artifacts from ERP systems”. In: *IEEE Transactions on Services Computing* 8.6 (2015), pp. 861–873 (cit. on p. 3).

[59] Artem Lutov, Mourad Khayati, and Philippe Cudré-Mauroux. “Accuracy evaluation of overlapping and multi-resolution clustering algorithms on large datasets”. In: *International Conference on Big Data and Smart Computing (BigComp)* (2019), pp. 1–8 (cit. on p. 171).

[60] Matin Mavaddat et al. “Facilitating business process discovery using email analysis”. In: *The First International Conference on Business Intelligence and Technology* (2011) (cit. on pp. 33, 35, 37).

[61] Dian Sa’adillah Maylawati and GA Putri Saptawati. “Set of Frequent Word Item sets as Feature Representation for Text with Indonesian Slang”. In: *Journal of Physics: Conference Series*. Vol. 801. 1. IOP Publishing. 2017, p. 012066 (cit. on p. 84).

[62] DS Maylawati. “The concept of frequent itemset mining for text”. In: *IOP Conference Series: Materials Science and Engineering*. Vol. 434. 1. IOP Publishing. 2018, p. 012043 (cit. on pp. 78, 84).

[63] DS Maylawati, H Aulawi, and MA Ramdhani. “The concept of sequential pattern mining for text”. In: *IOP Conference Series: Materials Science and Engineering*. Vol. 434. 1. IOP Publishing. 2018, p. 012042 (cit. on p. 84).

[64] Leland McInnes, John Healy, and Steve Astels. “hdbscan: Hierarchical density based clustering”. In: *The Journal of Open Source Software* 2.11 (2017), p. 205 (cit. on p. 32).

[65] Tom M Mitchell et al. “Extracting knowledge about users’ activities from raw workstation”. In: *AAAI* 1 (2006), p. 181 (cit. on pp. 6, 24, 26, 35–37).

[66] Business Process Model. “Notation (bpmn) version 2.0”. In: *OMG Specification, Object Management Group* (2011), pp. 22–31 (cit. on p. 29).

[67] Hamid Reza Motahari-Nezhad et al. “Event correlation for process discovery from web service interaction logs”. In: *The VLDB Journal* 20.3 (2011), pp. 417–444 (cit. on pp. 2, 20).

[68] Daniel Müllner. “Modern hierarchical, agglomerative clustering algorithms”. In: *arXiv preprint arXiv:1109.2378* (2011) (cit. on p. 96).

[69] E González López de Murillas, Hajo A Reijers, and Wil MP van der Aalst. “Case notion discovery and recommendation: automated event log building on databases”. In: *Knowledge and Information Systems* 62.7 (2020), pp. 2539–2575 (cit. on p. 20).

[70] Eduardo González López de Murillas, Hajo A. Reijers, and Wil M. P. van der Aalst. “Connecting databases with process mining: a meta model and toolset”. In: *Software and Systems Modeling* 18.2 (2019), pp. 1209–1247 (cit. on p. 21).

[71] Joyce Nakatumba and Wil MP van der Aalst. “Analyzing resource behavior using process mining”. In: *BPM*. Springer. 2009, pp. 69–80 (cit. on p. 3).

[72] Piotr Ożdżyński and Danuta Zakrzewska. “Using frequent pattern mining algorithms in text analysis”. In: *Information Systems in Management* 6 (2017) (cit. on p. 84).

[73] Jian Pei et al. “Mining sequential patterns by pattern-growth: The prefixspan approach”. In: *IEEE Transactions on knowledge and data engineering* 16.11 (2004), pp. 1424–1440 (cit. on p. 84).

[74] Airel Pérez-Suárez et al. “OClustR: A new graph-based algorithm for overlapping clustering”. In: *Neurocomputing* 121 (2013), pp. 234–247 (cit. on p. 149).

[75] Maja Pesic and Wil M. P. van der Aalst. “A Declarative Approach for Flexible Business Processes Management”. In: *BPM Workshops* 4103 (2006), pp. 169–180 (cit. on p. 52).

[76] Ashequl Qadir and Ellen Riloff. “Classifying sentences as speech acts in message board posts”. In: *EMNLP* (2011), pp. 748–758 (cit. on pp. 6, 10, 25, 27, 28, 35–37, 78).

[77] Eric K Ringger et al. “Task-focused summarization of email”. In: (2004) (cit. on pp. 6, 10, 25, 27, 28, 35, 37, 78, 192).

[78] Andreas Rogge-Solti. “Probabilistic Estimation of Unobserved Process Events”. PhD thesis. Universität Potsdam, 2014 (cit. on pp. 2, 20).

[79] Andreas Rogge-Solti et al. “Repairing event logs using timed process models”. In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer. 2013, pp. 705–708 (cit. on p. 20).

[80] Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *ArXiv* abs/1910.01108 (2019) (cit. on pp. 116, 204).

[81] Stefan Schönig et al. “Mining team compositions for collaborative work in business processes”. In: *SoSyM* 17.2 (2018), pp. 675–693 (cit. on pp. 3, 22).

[82] John R Searle. “A taxonomy of illocutionary acts”. In: (1975) (cit. on p. 28).

[83] John R Searle and John Rogers Searle. *Speech acts: An essay in the philosophy of language*. Vol. 626. 1969 (cit. on pp. 28, 33).

[84] Leslie Shing et al. “Extracting workflows from natural language documents: A first step”. In: *International Conference on Business Process Management* (2018), pp. 294–300 (cit. on p. 34).

[85] Mark S Silver, M Lynne Markus, and Cynthia Mathis Beath. “The information technology interaction model: A foundation for the MBA core course”. In: *MIS quarterly* (1995), pp. 361–390 (cit. on p. 1).

[86] Diego Carvalho Soares, Flávia Maria Santoro, and Fernanda Araujo Baião. “Discovering collaborative knowledge-intensive processes through e-mail mining”. In: *Journal of Network and Computer Applications* 36.6 (2013), pp. 1451–1465 (cit. on pp. 6, 10, 25, 27–29, 35–37, 78, 148).

[87] Minseok Song and Wil MP Van der Aalst. “Towards comprehensive support for organizational mining”. In: *Decision support systems* 46.1 (2008), pp. 300–317 (cit. on p. 3).

[88] Mohammad S Sorower. “A literature survey on algorithms for multi-label learning”. In: () (cit. on p. 173).

[89] Joseph Valacich and Christoph Schneider. *Information systems today: Managing in the digital world*. Prentice Hall, 2012 (cit. on p. 1).

[90] Wil Van Der Aalst. "Data science in action". In: *Process mining*. Springer, 2016, pp. 3–23 (cit. on pp. 2, 3).

[91] Wil Van Der Aalst et al. "Process mining manifesto". In: *International Conference on Business Process Management* (2011), pp. 169–194 (cit. on pp. 2, 18, 19).

[92] Wil MP Van Der Aalst and Maja Pesic. "DecSerFlow: Towards a truly declarative service flow language". In: *International workshop on web services and formal methods*. Springer. 2006, pp. 1–23 (cit. on p. 21).

[93] Wil MP Van Der Aalst, Hajo A Reijers, and Minseok Song. "Discovering social networks from event logs". In: *CSCW* 14.6 (2005), pp. 549–593 (cit. on p. 3).

[94] Boudewijn F Van Dongen et al. "The ProM framework: A new era in process mining tool support". In: *International conference on application and theory of petri nets*. Springer. 2005, pp. 444–454 (cit. on p. 21).

[95] HMW Verbeek et al. "Xes, xesame, and prom 6". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2010, pp. 60–75 (cit. on p. 21).

[96] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. "Process mining with the heuristics miner-algorithm". In: *Technische Universiteit Eindhoven, Tech. Rep. WP* 166 (2006), pp. 1–34 (cit. on pp. 3, 10, 21, 148).

[97] K-L Wu, Philip S. Yu, and Allen Ballman. "Speedtracer: A web usage mining and analysis tool". In: *IBM Systems Journal* 37.1 (1998), pp. 89–105 (cit. on p. 20).

[98] Zhi-Qiang Zeng et al. "Fast training support vector machines using parallel sequential minimal optimization". In: *3rd international conference on intelligent system and knowledge engineering* 1 (2008), pp. 997–1001 (cit. on p. 25).

[99] Jie Zhang and Ali A Ghorbani. "The reconstruction of user sessions from a server log using improved time-oriented heuristics". In: *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.* IEEE. 2004, pp. 315–322 (cit. on pp. 2, 20).

DOCTORAL SCHOOL

Titre: Découverte des processus métiers à partir des Emails, un premier pas vers la gestion des processus métiers dans des systèmes d'information moins structurés

Mots clés: Processus Métiers, Courriels, Système d'Informations, Gestion des Processus Métiers

Résumé: La fouille de processus vise à analyser les traces d'exécution des systèmes d'information (SI), utilisés dans le cadre des activités métiers, pour découvrir des connaissances sur les processus métiers (PM). D'importants travaux de recherche ont été menés dans ce domaine. Cependant, ils supposent généralement que ces traces d'exécution ont un niveau de structuration élevé. Cela signifie que: (i) ils sont composés d'enregistrements structurés, chacun capturant l'exécution d'une activité, et (ii) une partie des attributs des événements d'exécution (comme le nom de l'activité, l'horodatage) sont explicitement inclus dans ces enregistrements, ce qui facilite leur inférence. Néanmoins, les PM peuvent être entièrement ou partiellement réalisés dans des SI moins structurés générant des traces d'exécution de faible niveau de structuration. Les systèmes de courriels sont largement utilisés pour réaliser de manière collaborative des activités de PM. Cependant, leurs traces d'exécution sont de nature non-structurée de point de vue découverte des PM, ce qui empêche l'application directe des techniques existantes. Pour celles qui découvrent les PM à partir des courriels, elles: (i) nécessitent généralement une intervention humaine, et (ii) se sont limitées à la découverte des PM selon la perspective comportementale.
Dans cette thèse, nous proposons de découvrir des fragments de PM à partir des courriels selon leurs perspectives fonctionnelles, données, organisationnelles et comportementales. Nous formalisons d'abord ces perspectives en considérant les spécificités des systèmes de courriels. Nous introduisons la notion de contribution des acteurs à la réalisation des activités pour enrichir les perspectives organisationnelles et comportementales. Nous considérons en outre les entités informationnelles manipulées par les activités de PM pour décrire la perspective des données. Pour automatiser la découverte de l'ensemble des perspectives, nous introduisons une approche complètement non-supervisée. Cette approche transforme principalement les traces non structurées des courriels en un journal d'événements structuré avant de l'analyser pour découvrir les PM selon différentes perspectives. Nous introduisons dans ce contexte un ensemble de solutions algorithmiques pour: (i) l'apprentissage non supervisé des activités basé sur la découverte de motifs fréquents de mots dans les courriels, (ii) la découverte des occurrences des activités dans les emails pour capturer les attributs des événements, (iii) la découverte des actes de parole des expéditeurs pour reconnaître leurs intentions de mentionner les activités dans les emails afin de déduire leurs contributions dans leur réalisation, (iv) le regroupement par chevauchement des activités pour découvrir leurs artefacts manipulés (c.-à-d. les entités informationnelles), et (v) la découverte des contraintes séquentielles entre les types d'événements pour découvrir la perspective comportementale des PM.
Notre approche est validée en utilisant des courriels publics d'Enron. Nos résultats sont en outre rendus publics pour permettre des comparaisons quantitatives avec des travaux connexes s'ils utilisent le même ensemble de données publiques. Nous montrons enfin l'utilité de nos résultats pour améliorer la gestion des PM à travers deux applications: (i) un outil de découverte et de recommandation des connaissances de PM à intégrer dans un système de gestion de courriels, et (ii) l'analyse de données CRM pour l'exploration des raisons de la satisfaction/non-satisfaction des utilisateurs.

Title: Business process discovery from emails, a first step towards business process management in less structured information systems

Keywords: Business Process, Emails, Information Systems, Business Process Management

Abstract: Process discovery aims at analysing the execution logs of information systems (IS), used when performing business activities, for discovering business process (BP) knowledge. Significant research works has been conducted in such area. However, they generally assume that these execution logs are of high or of middle level of maturity w.r.t BP discovery. This means that (i) they are composed of structured records while each one captures evidence of one activity execution, and (ii) a part of events' attributes (e.g. activity name, timestamp) are explicitly included in these records which facilitates their inference. Nevertheless, BP can be entirely or partially performed through less structured IS generating execution logs of low level of maturity. More precisely, emailing systems are widely used as an alternative tool to collaboratively perform BP tasks. Traditional BP discovery techniques could not be applied or at least not directly applied due to the unstructured nature of email logs data. Recently, there have been several initiatives to extend the scope of BP discovery to consider email logs. However, most of them: (i) mostly require human intervention, and (ii) were limited to BP discovery according to its behavioral perspective.
In this thesis, we propose to discover BP fragments from email logs w.r.t their functional, data, organizational and behavioral perspectives. We first formalize these perspectives considering emailing systems specifities. We introduce the notion of actors' contributions towards performing activities to enrich the organizational and the behavioral perspectives. We additionally consider the informational entities manipulated by BP activities to describe the data perspective. To automate their discovery, we introduce a completely unsupervised approach. This approach mainly transforms the unstructured email log into a structured event log before mining it for discovering BP w.r.t multiple perspectives. We introduce in this context several algorithmic solutions for: (i) unsupervised learning activities based on discovering frequent patterns of words from emails, (ii) discovering activity occurrences in emails for capturing event attributes, (iii) discovering speech acts of activity occurrences for recognizing the sender purposes of including activities in emails, (iv) overlapping clustering of activities to discover their manipulated artifacts (i.e. informational entities), and (v) mining sequencing constraints between event types to discover BP behavioral perspective.
We validated our approach using emails from the public dataset Enron to show the effectiveness of the obtained results. We publicly provide these results to make quantitative comparisons feasible with related works if they use the same public dataset. We finally show the usefulness of our results for improving BPM through two potential applications: (i) a BP discovery & recommendation tool to be integrated in emailing systems, and (ii) CRM data analysis for mining reasons of users' satisfaction/non-satisfaction.

Institut Polytechnique de Paris

91120 Palaiseau, France