Standardization of textual data for comprehensive job market analysis

Emmanuel Malherbe

NNT : 2016SACLC058

Thèse de doctorat de l'Université Paris-Saclay préparée CentraleSupélec

Ecole doctorale n°573
INTERFACES :
Approches Interdisciplinaires, Fondements, Applications et Innovation

Spécialité de doctorat: Informatique

par

M. Emmanuel Malherbe

Normalisation Textuelle pour une Analyse Exhaustive du Marché de l'Emploi

Thèse présentée et soutenue à Châtenay-Malabry, le 18 novembre 2016.

Composition du Jury :

M.	Antoine Cornuéjols	Professeur AgroParisTech	(Président du jury)
Mme	Pascale Kuntz	Professeur Université de Nantes	(Rapporteur)
M.	Younès Bennani	Professeur Université Sorbonne Paris Cité	(Rapporteur)
M.	Jean-Baptiste Gariel	Ingénieur R&D Multiposting	(Examinateur)
Mme	Marie-Aude Aufaure	Professeur CentraleSupélec	(Directrice de thèse)
M.	Mario Cataldi	Maître de Conférences Université Paris VIII	(Co-encadrant de thèse)

Abstract

With so many job adverts and candidate profiles available online, the e-recruitment constitutes a rich object of study. Large-scale analysis of this data would enable us to fully understand the offers and demands of the work force. All this information is however textual data, which from a computational point of view is unstructured. The large number and heterogeneity of recruitment websites also means that there is a lot of vocabularies and nomenclatures. A French firm specialized in e-recruitment tools, Multiposting, that wanted to easily manipulate this data, financially supported the research of this thesis as well as provided access to millions of CVs and job ads aggregated from public sources.

One of the difficulties when dealing with this type of textual data is being able to grasp the concepts contained in it, since the meaning behind the words can only be completely understood by humans. Inferring structured and meaningful attributes from raw textual data, i.e. standardization, is the problem that is tackled in this thesis. The aim of standardization is to create a unified process providing values in a nomenclature. A nomenclature is by definition a finite set of meaningful concepts, which means that the attributes resulting from standardization are a structured representation of the information. This process thus translates each document into a common language, which allows the aggregation of all data in an understandable and exploitable way. Several questions are however raised: Are the websites' structured data usable for a unified standardization? What structure of nomenclature is the best suited for standardization, and how to leverage it? Is it possible to automatically build such a nomenclature from scratch, or to manage the standardization process without one?

To illustrate the various obstacles of standardization, the examples we are going to study include the inference of the skills or the category of a job advert, or the level of training of a candidate profile. One of the challenges of e-recruitment is that the concepts are continuously evolving, which means that the standardization must be up-to-date with job market trends. In light of this, we will propose a set of machine learning models that require minimal supervision and can easily adapt to the evolution of the nomenclatures. The questions raised found partial answers using Case Based Reasoning, semi-supervised Learning-to-Rank, latent variable models, and leveraging the evolving sources of the semantic web and social media. The different models proposed have been tested on real-world data, before being implemented in a industrial environment. The resulting standardization is at the core of SmartSearch, a project which provides a comprehensive analysis of the job market.

Résumé

Sachant qu'une grande partie des offres d'emplois et des profils candidats est en ligne, le e-recrutement constitue un riche objet d'étude. L'analyse à grande échelle de ces données permettrait notamment de fluidifier le marché du travail. Ces documents sont cependant des textes non structurés, et le grand nombre ainsi que l'hétérogénéité des sites de recrutement implique une profusion de vocabulaires et nomenclatures. Avec l'objectif de manipuler plus aisément ces données, Multiposting, une entreprise française spécialisée dans les outils de e-recrutement, a soutenu cette thèse, notamment en terme de données, en fournissant des millions de CV et offres d'emplois agrégées de sources publiques.

Une difficulté lors de la manipulation de telles données est d'en déduire les concepts sous-jacents, les concepts derrière les mots n'étant compréhensibles que des humains. Déduire de tels attributs structurés à partir de donnée textuelle brute est le problème abordé dans cette thèse, sous le nom de normalisation. Avec l'objectif d'un traitement unifié, la normalisation doit fournir des valeurs dans une nomenclature, de sorte que les attributs résultants forment une représentation structurée unique de l'information. Ce traitement traduit donc chaque document en un language commun, ce qui permet d'agréger l'ensemble des données dans un format exploitable et compréhensible. Plusieurs questions sont cependant soulevées : peut-on exploiter les structures locales des sites web dans l'objectif d'une normalisation finale unifiée ? Quelle structure de nomenclature est la plus adaptée à la normalisation, et comment l'exploiter ? Est-il possible de construire automatiquement une telle nomenclature de zéro, ou de normaliser sans en avoir une ?

Pour illustrer le problème de la normalisation, nous allons étudier par exemple la déduction des compétences ou de la catégorie professionelle d'une offre d'emploi, ou encore du niveau d'étude d'un profil de candidat. Un défi du e-recrutement est que les concepts évoluent continuellement, de sorte que la normalisation se doit de suivre les tendances du marché. A la lumière de cela, nous allons proposer un ensemble de modèles d'apprentissage statistique nécessitant le minimum de supervision et facilement adaptables à l'évolution des nomenclatures. Les questions posées ont trouvé des solutions dans le raisonnement à partir de cas, le learning-to-rank semi-supervisé, les modèles à variable latente, ainsi qu'en bénéficiant de l'Open Data et des médias sociaux. Les différents modèles proposés ont été expérimentés sur des données réelles, avant d'être implémentés industriellement. La normalisation résultante est au coeur de SmartSearch, un projet qui fournit une analyse exhaustive du marché de l'emploi.

Remerciements

Mes premiers remerciements vont évidemment à Marie-Aude Aufaure, ma tutrice, sans qui cette thèse n'aurait jamais eu lieu. Bien qu'arrivé en cours de route, j'adresse aussi un chaleureux merci à Mario, qui a su endosser avec brio le rôle de co-encadrant. Nos rencontres et tes conseils ont été très précieux pour moi, d'autant plus que tu as toujours montré une bonne humeur et un soutien moral. C'est avec une certaine nostalgie que je remercie le reste de l'ex-équipe Business Intelligence, à savoir Yves, Nesrine, Delphine, Thomas et Marine, qui ont chacun aidé à cette thèse, par le biais de collaborations, relectures et discussions. Je remercie aussi Mamadou, pour ta bonne humeur et notre collaboration, car tu m'a montré l'exemple avec ton année d'avance sur l'aventure d'une thèse CIFRE en startup web.

Je tiens à remercier les rapporteurs de cette thèse, Pascale Kuntz et Younès Bennani, pour leur relecture et remarques avisés. Je remercie aussi Antoine Cornuéjols pour avoir accepté de faire partie du jury de cette thèse.

Du côté de Multiposting, je dois évidemment remercier amicalement Clément, Shuai, JB, Loïc et Laert. Tout d'abord pour leur relecture et leurs conseils ; mais aussi pour tout le soutien ainsi que les bons moments qu'ils m'ont apportés au cours de ces trois ans, accompagnés aussi de Manu, Séb et Benjamin. Katarina, tu mérites une mention spéciale pour ta relecture et ton investissement pour améliorer mon anglais - là ou Google Translate ne pouvais plus rien pour moi. Un grand merci aussi à Camille, dont la relecture a aussi grandement amélioré l'anglais de ce manuscrit.

Deux personnes ont aussi beaucoup aidé à cette thèse, sans vraiment le savoir : il s'agit de Nguyen et Aude. Je dois vous remercier tout particulièrement car sans la perspective de vous voir, j'aurais été bien moins assidu à Centrale ; vous avez donc indirectement mais profondément aidé à ce que mon travail de rédaction reste régulier. Afin de remercier là encore des êtres qui me sont chers, j'adresse un merci à Alexis, Sylvain, Claire et Christophe, qui ont su être présents quand il le fallait. Et à nouveau, Camille, merci pour tout ce que tu m'as apporté sur ces deux dernières années. Soulage-toi, j'arrêterai de te peser sur le moral avec mes histoires de thêsard...

Contents

Chapter 1

Introduction and Motivation

1.1 The Internet, a Rich but Unstructured World

The growth of the internet has had a similar impact on our daily lives as the industrial revolutions. It has created new jobs and businesses, changed our leisure activities and facilitated worldwide communication. The internet is now synonymous with a huge amount of data, including videos, images, but also text: most webpages are text in natural language, and represent an incredibly rich source of information, for instance Wikipedia is a multilingual encyclopedia that explains millions of concepts. New fields of research have emerged, most of them very promising and exciting, with the aim of improving the user experience of web applications and making use of this rich source of data.

The most common examples of internet-oriented research efforts are information retrieval systems, and recommender systems. Information retrieval systems find documents related to a specific query, and are typically used in search engines. Recommender systems suggest items to a user, based on his/her preferences, and are typically used on shopping websites. Both types of system are often assisted by machine learning, which enables the automatic reproduction of a behaviour from a base of examples. These systems often have to manage textual data, which is mainly unstructured. Some models use calculations that do not take word order into account. These models gives good results for specific tasks, such as retrieving relevant documents for an unstructured query. However, they miss the underlying concepts behind the words. These structured concepts are more important since they can be used as input features for machine learning and artificial intelligence.

One current direction of research for capturing the meaning behind words focuses on organizing them in a graph of concepts. This type of graph constitutes a knowledge base, and is structured thanks to a formal naming of the edges and types, which is called an ontology. An example of a generic knowledge base is WordNet, which is an advanced dictionary that aims to model natural language in its entirety. In particular, WordNet does not treat specific entities or concepts such as companies and job categories. For specific domains, building a knowledge base from scratch and ensuring its maintenance incurs high engineering costs, because it requires a lot of manual work. Fortunately, some high quality knowledge

bases can be found on the Internet, sometimes in the form of *nomenclatures* with meta-data, i.e. lists of concepts with associated information. If these nomenclatures do not allow the extraction of concepts from a text, they are nevertheless well described and maintained on a regular basis. If we are able to associate a document to the corresponding concept(s) of a nomenclature, this means that we have partly understood it, and can lead to more advanced computations. When dealing with several textual documents, such an association is a standardization in the sense that each document is explained in the same language, namely the nomenclature. Furthermore, when the nomenclature is hierarchized and has meta data associated to it, the standardized documents benefit from this additional knowledge. The difficulty remains on how to associate a concept to a textual document, and currently no generic approach exists.

While advances in research related to artificial intelligence have already benefited e-commerce, e-recruitment has gained little from these advances, despite the big stakes associated to it. One of the challenges related to this field is that it involves many concepts and is constantly evolving. This will be the object of study throughout this thesis.

1.2 Industrial Context

Before introducing the reader to the e-recruitment topic, we would first like to describe the processes involved in recruitment. This will better motivate the SmartSearch project, which is currently being developed by Multiposting and strongly relies on the work of this thesis.

1.2.1 From Recruitment to e-Recruitment

Recruiting means finding someone for a vacant position, for example a cashier for a small shop, a farmer, or a computer scientist for an IT company. We could say that recruitment is one of the oldest activities of mankind, since it appeared at the same time as the concept of work. The first explicit trace of a coherent recruitment strategy dates back to the roman empire [Losey, 1998], when Julius Caesar proposed a reward of 300 sestertii to any soldier that would convince another man to join the roman army. Centuries have passed and recruitment has spread to all job sectors, and today, there are even employees dedicated to this activity, recruiters. This process is indeed a difficult task, and requires a lot of time.

In this thesis, we will take on the recruiter's point of view, and not that of the candidate looking for a new position. To describe the task of recruiting an employee, we propose to distinguish 4 major steps:

The analysis: preliminary to any action, the recruiter needs to know about the job market, what type of profile could fit the position, and where to find it.

The sourcing: the recruiter creates a pool of candidates. It can be passive, meaning that a job advert is made public, and the recruiter waits for

candidates to apply. But, it can also be active, meaning that the recruiter directly searches for suitable candidates. We can use a metaphor to make this clearer: in the first case, the recruiter fishes for candidates using the advert as bait, and in the second case he hunts for them.

The selection: from his/her pool of candidates, the recruiter only keeps the relevant candidates.

The interview: the recruiter meets each of the suitable candidates, in order to see if the person fits the position. Once this step has been completed, a candidate is hired.

The interview is a necessary step and has to be done by the recruiter; this is a very human task, and can involve some negotiation, such as for the salary. As this last step is susceptible to failing, the aim of sourcing and selection processes is to find the maximum number of suitable candidates. Unlike the interview step, the first three steps can easily be dematerialized, and are thus subject to change with the breakthrough of computer science. Nowadays, for instance, sourcing is almost exclusively done on the internet, whereas during the 20th century it was mainly done by adverts in newspapers. Today there are plenty of websites dedicated to recruitment; for passive sourcing, the recruiter can post job adverts on websites called job boards. Some websites such as *Indeed* and *CareerBuilder* do not have a specific audience, others, such as *Efinancialcareers* are specialized in a specific sector or job category. For active sourcing the recruiter looks for candidate profiles in CV-Banks, like *Monster*, as well as in professional social networks, such as *LinkedIn* or *Viadeo*. Candidates can make their resume available online, but professional networks are now more commonly used and also enable recruiters to see candidate connections. This type of sourcing using a numerical support and used widely is known as e-recruitment.

The consequences of the digitization of recruitment are multiple. Firstly, communication has been facilitated, millions of jobs can be accessed by candidates, and recruiters can also access millions of profiles online. Secondly, this new way of sourcing candidates enables the recruiters to concentrate on the more human aspects of recruitment, namely the interview. Lastly, the whole job market is represented by this numeric data which can be used for large scale computing, and in particular has opened the way for a computer assisted recruitment process.

1.2.2 SmartSearch Project: Unifying Job Market Information

With the digitization of recruitment, many new businesses emerged, including recruitment websites, such as job boards, CV banks and professional social networks. In this context, Multiposting has for objective is to assist and ease the work of recruiters. This French firm that was founded in 2008 and recently acquired by SAP; it has now more than 80 employees and thousands of customers. The firm has developed several products that help the recruiter in the sourcing process, including a tool that enables recruiter to post jobs to multiple channels on the Internet.

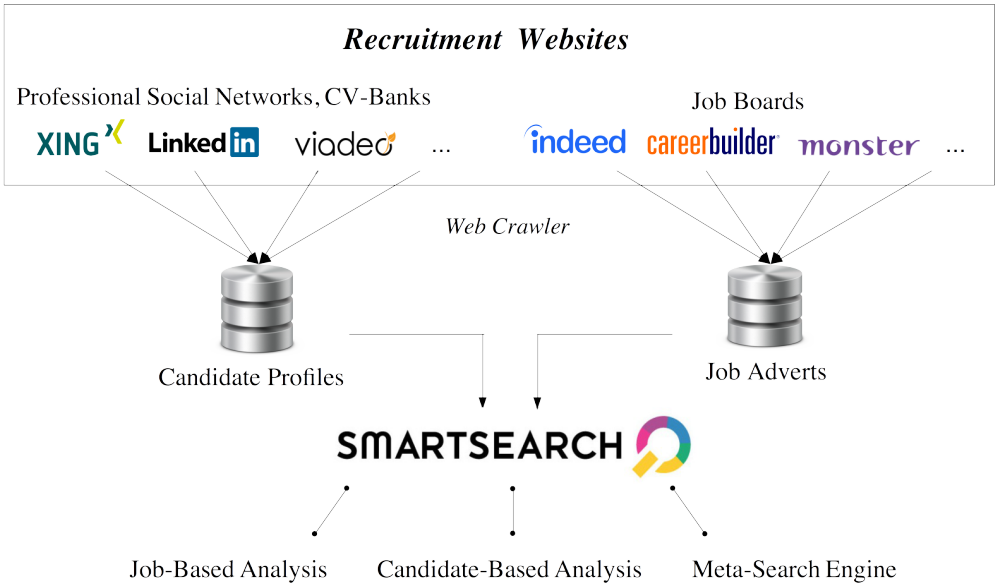


Figure 1.1 – The overall functioning of SmartSearch, which leverages internet data to provide comprehensive job market analysis.

The research effort of the company is now focused on SmartSearch. This project is currently under development and has been laureate on two occasions in the "Big Data" category of the national Concours Mondial de l'Innovation[1], which awards the most innovative industrial projects in France. The project has also been supported by the Ile-de-France region through the research partnership SONAR[2], which enabled Multiposting to collaborate with the LIASD[3] and the LIMSI[4], the former an IUT Montreuil laboratory specialized in statistics and the latter a CNRS laboratory specialized in semantics. SmartSearch has also benefited from the results of this thesis, which is supported by the ANRT[5] through the CIFRE program.

The aim of SmartSearch is to aggregate all the job market data available on recruitment websites, and to then propose comprehensive statistics about it. As shown in Figure 1.1, the data used includes jobs, taken from job boards, and numeric resumes, taken from CV Banks or professional social networks. This data is processed and stored in a large database, and then used to compute different statistics, with the following purposes:

- Job-based job market analysis: These statistics are computed from the jobs. For instance, given a job category, the recruiter can access information

[1] `http://www.gouvernement.fr/argumentaire/concours-mondial-de-l-innovation`
[2] `http://sonar-project.com/`
[3] Laboratoire d'Informatique Avancée de Saint-Denis: `http://www.ai.univ-paris8.fr/`
[4] Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur: `https://www.limsi.fr/`
[5] Association Nationale de la Recherche et de la Technologie: `www.anrt.asso.fr`

about which companies are hiring the most, suitable recruitment websites, the average required experience or where the most job openings are located.

- Candidate-based job market analysis: These statistics are computed from the candidate profiles. For instance, given a skill, recruiters can view relevant educational institutions, or companies that have the most employees for a particular competency.
- Meta-search engine for jobs and candidates: The recruiters can look at practical examples of jobs or candidates related to a specific industry, institution or skill. This search engine aims to support the computed statistics by showing real-world examples.

SmartSearch is consequently the ideal tool for the first step of the recruitment process, namely the analysis step. It also provides critical information for job sourcing, by showing suitable candidate profiles or suggesting relevant recruitment websites. The computation involved in the job sourcing analysis could also be used for the selection step of the recruitment process. However, the project does not focus on the job/candidate matching but aims to give the most comprehensive and updated snapshot of the job market.

	Resumes	Jobs
Number of entries	4,352,948	9,521,239
Number of sources	42	120
Average number of fields per source	35.1	10.2

Table 1.1 – Statistics of numbers of documents used in the SmartSearch project.

For the project, an engine continuously collects data from many different websites. The number of websites involved at the time of the writing of the thesis is displayed in table 1.1, and it is planned to keep increasing. On each website, the job is made up of several fields, such as the contract type, the title, and the description. The structure and the content of the fields can be very different according to the source. The average number of fields is shown in table 1.1. The computations involved in SmartSearch require every job and every profile to be translated into a common *constructed language*, because of the heterogeneity of the sources. This problem is referred to as interoperability, and is a common issue when using internet data [Ouksel and Sheth, 1999]. The translation aims to give each job advert the same structure of fields, as well as represent the content of the fields in a singular way. In the next section, we detail the field structure of jobs and candidate profiles, which aims to be generic.

1.3 The Object of our Work: Textual Data

In this section the generic fields that compose jobs and candidate profiles are described. These fields are also found in SmartSearch documents. Both templates detailed below aim to be as generic as possible and can be found on most recruitment websites, even if occasionally some fields are missing.

1.3.1 Job Adverts

Generally speaking, a job advert is a document which describes a vacant position. This document is posted on recruitment websites, and is written to be as attractive as possible, in order to catch the candidate's eye. The job advert is separated into several textual fields, for which we propose the following generic structure:

Title : Describes the job position in a couple of words.

Description : Describes the job in a couple of sentences, such as the tasks to be performed by the hired candidate and the team that he would join.

Company description : Describes the company in a few sentences, such as its sector and corporate spirit.

Profile : Describes the profile is required for the position in a few sentences, such as the education level, experience and skills necessary or nice to have.

Company : The name of the firm that is hiring.

Location : The city where the employee will work.

Contract : The type of contract, for example short term contract or part-time work.

Study : The level of study required for the position.

Experience : The amount of experience required in the sector.

Sector : The sector of the company.

Category : The job category of the position, generally coarse-grained.

Each of the fields above can possibly be empty, if the recruiter has not provided all the information, or if the job advert is published on a website which does not include the field.

1.3.2 Candidate Profiles

A candidate is represented by a resume or a user profile on a professional social network. Since the emergence of the CV, a common template is widely used, with some variations. We consider that a profile can be described by using the following groups of fields:

Education : This group of fields describes the degrees obtained by the candidate. Each degree has a *Start date*, *End date*, *Degree name*, *Educational Institution* and *Location*.

Positions : This group of fields describes the positions previously occupied by the candidate. Each job position is similar to a job advert as described above, but shorter and with only the *Title*, *Description* and *Company*, *Start date*, *End date* and *Location*.

Skills : This group describes the candidate's different skills.

The last field is generally filled by two different type of skills, hard skills and soft skills. A hard skill is precise and easily quantifiable, for example the ability to use a specific tool or software. On the contrary, a soft skill refers to a personality trait, such as leadership, empathy or the communication skills. In our representation of the candidate profile, we omitted additional information such as interests, campus activities or volunteer experiences. Although this information is sometimes helpful for giving an insight into candidate personality traits which are important for a particular role [Cole et al., 2007], it is difficult for a recruiter to infer such traits just from a written résumé [Cole et al., 2009]. We also omit any personal information such as name, phone number, email address, for confidential reasons and also because they are irrelevant for our study.

1.3.3 Structured or Unstructured Data?

All the fields described above are textual, in the sense that they are stored as text, and displayed as verbatim to the user. However, it is necessary to distinguish two types of fields. The first type, ***unstructured*** fields, are purely textual: they are generally written in natural language and have an infinite number of possible values, such as a job *Description*, *Title* or a *Degree Name*. ***Structured*** fields are ones that have a finite number of possible values, such as the *Contract* or the required *Experience* of a job advert. In practice, the value of a structured field is specified by the user when selecting an option in a *nomenclature*, which is a list of concepts. Every website has its own nomenclature for a given structured field, meaning that the field would be represented differently depending on the website. Some fields, such as company field and educational institutions field, are only structured on some websites. For the purpose of this thesis, we will consider that every field that could be associated to a nomenclature is structured, meaning that there are a finite number of options that could be listed.

In this thesis, the values of a nomenclature will be called the concepts. Two types of nomenclature can be distinguished:

Nomenclature of entities : this is a set of concepts that exist, called entities. Each entity has one or several names, and if it is not explicitly named in the document, it can not be inferred from it. For instance, a *Company*, an *Institution*, a *Location*, a *Skill* are examples of entities. Nomenclatures can be very large, because they list all the possible entities.

Abstract Nomenclature : this is a set of high-level concepts, such as the *Category* or the *Sector* field. Contrary to entities, we can not associate a high-level concept to an exact term or name. As a consequence, the link between a document and such a concept is often unclear, like the *Category* in a jobadvert description and a *Study Level* in a degree name. The concept is implicit in the text, and therefore needs to be inferred. This type of nomenclatures is generally shorter than a nomenclature of entities, and never exceeds one thousand concepts.

By definition, the concepts are often associated to some textual data, such as a label. This meta data provides interesting knowledge, and will be regularly

used in the models proposed throughout the thesis. To illustrate the different reasoning, we will invent a character called Bob.

1.4 Have you met Bob?

Before manipulating the data described above, we would like to introduce the reader to Bob. This fictional character will accompany the reader throughout the thesis and will illustrate problems in a simple way.

Bob is very nice and does what he is told. This docility is crucial, because Bob is the only person that directly manipulates the numeric data. In practice we just give him orders, and Bob applies them carefully when manipulating the job adverts or CVs. In fact, Bob suffers from a cruel lack of autonomy, which is due to a simple fact: Bob is not very smart. In particular, he does not understand what he reads. He can read, yes; but he can not understand the meaning of words.

Consequently, Bob can handle *structured* fields much better than purely textual ones. In order to deal with Bob's limited ability, we have to give him particularly precise and detailed rules. Aside from this Bob has several strengths: he can count, he executes orders extremely quickly, and has an almost unlimited memory capacity.

This curious example is not only designed to amuse the reader, but also has a real meaning. The name Bob is not a nickname for Robert, but for *Robot*. Our new friend represents what is known as artificial intelligence, even if he is not so smart at the moment. However, chapter after chapter we will endeavor to teach Bob the best way possible and make him as artificially intelligent as possible.

1.5 The Need for Generic Standardization

The textual data described in Section 1.3 partly consists of unstructured fields, that can not be understood by Bob. Some calculations only consider the words contained in the texts, but we want to go beyond words for structured concepts: this means that we need to convert the textual data into structured information. In other words, given a text, we would like to infer a structured attribute, which would be a value chosen in a nomenclature. The choice of concept to be inferred depends on the use case, and is not discussed in this thesis; for instance, we may want to infer the job category from the textual fields of the job advert, or the official name of the institution that a candidate graduated from, using the "education" field of his/her resume. We call this *standardization*, and it is the problem we address in this thesis:

"How can we link an unstructured textual document to the correct concept in a nomenclature, in a cheap and updatable manner?"

For each field that we need to standardize, two questions must be answered:

how to determine the correct nomenclature - possibly by generating it automatically - and how to establish a strategy for finding the concept that correctly corresponds to a piece of textual data. It should be noted that standardization does not lead to a gain of information, since structured data can not convey more information that what is contained in the original text. The concept represented by the nomenclature can correspond exactly to the textual information - for example the official name for an educational institution - or can be higher-level - such as the category of a job advert - and in this case it conveys less information than the initial text.

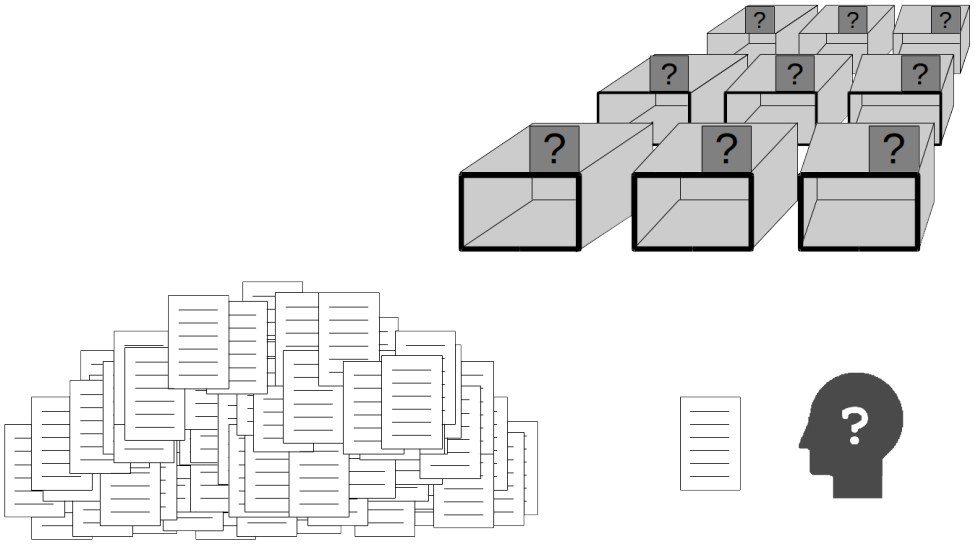


Figure 1.2 – The problem of textual data standardization from Bob's point of view: each paper has to fit into one box, a piece of paper represents a document and a box represents a concept. All of these concepts (represented by each box) grouped together represent the nomenclature.

From Bob's point of view, all that the standardization process requires him to do is put a sheet of paper into a box, as shown in Figure 1.2. The document contained on the sheet of paper is the unstructured textual data, and each box represents a concept of the nomenclature. We need to determine what set of boxes to take into consideration, and to tell Bob how to decide in which box he should put a given paper. Fortunately Bob is fast and once he is told what to do, he can process a large number of pieces of paper in a very short amount of time. Moreover, he can easily work out basic statistics according to the boxes, such as the number of documents that are contained in a box.

We need to be more precise about some aspects of our definition of standardization. Firstly, standardization is not only necessary for unstructured data, but also for structured data, when it as aggregated from several websites. Indeed when we deal with multiple sources, the data is structured into different nomenclatures according to each website, which means that the same concept will be expressed using different words, and possibly in many different ways.

This prevents us from being able to compare data from two different sources. In this case, standardizing signifies the conversion of all values into a singular nomenclature, in order to have a more global perspective in comparison with data represented in the original nomenclatures. Secondly, in some cases - as with structured data - the documents that are inputed convey exactly the same information as the nomenclature. In other cases, standardization provides different information from that contained in the original document, such as when determining the study level of an educational degree. Thirdly, several textual fields can be standardized at once. For instance, when we infer the category of a job advert, the inputed data will be all of the advert's unstructured fields. Lastly, the same document can be subject to multiple standardizations, meaning we can infer several different concepts from it: for instance, we can obtain the industrial sector and the job category from the same job advert.

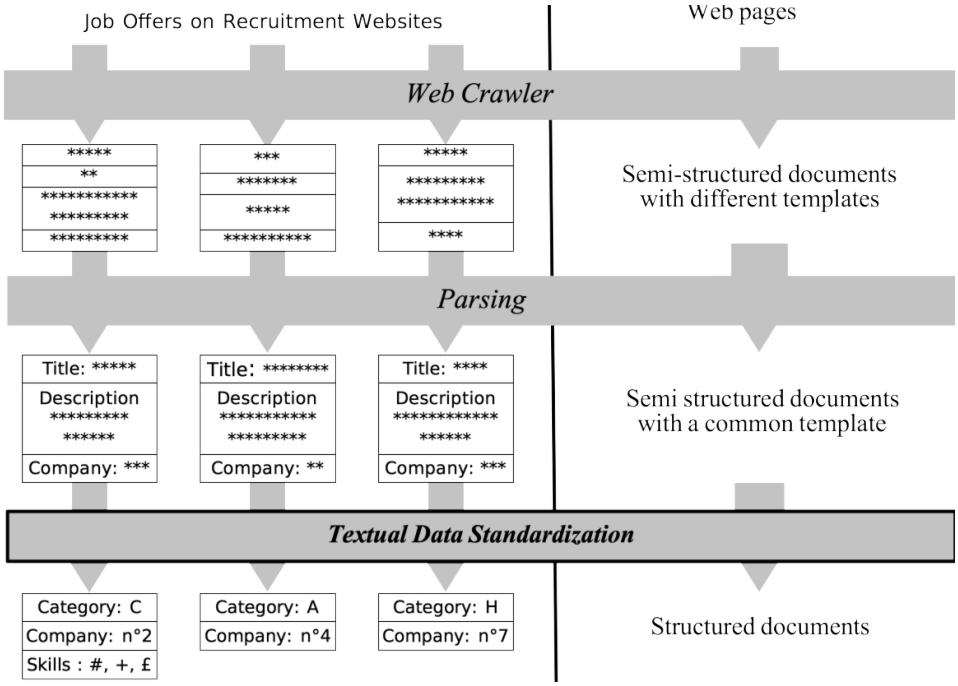


Figure 1.3 – Documents are processed as raw internet webpages into a structured database. The example of SmartSearch job adverts is on the left hand side.

Standardization is a crucial part of the SmartSearch project, and it is used as a pre-process when we fill the database with job adverts and candidate profiles, as shown in the diagram of Figure 1.3. The documents collected from the internet have a field structure that depends on each website. This structure is automatically aligned to the structure of Section 1.3, using a similar process to [Naz, 2009] that will not be discussed in this thesis. The documents are therefore organized according to the same textual fields before the standardization process, so that we can then leverage this common template in the standardization process.

Having standardized the job adverts and candidate profiles in SmartSearch's

database, we are then able to easily answer questions such as "What are the most sought-after skills for Sales managers?" or "What are the study levels and educational institutions of computer scientists in Paris?". Indeed, such questions can not be understood by just looking at the words contained in each text, but are expressed in the structured fields. Standardization is therefore crucial when dealing with textual data. The next section details our strategy for tackling this problem.

1.6 Structure of the Thesis

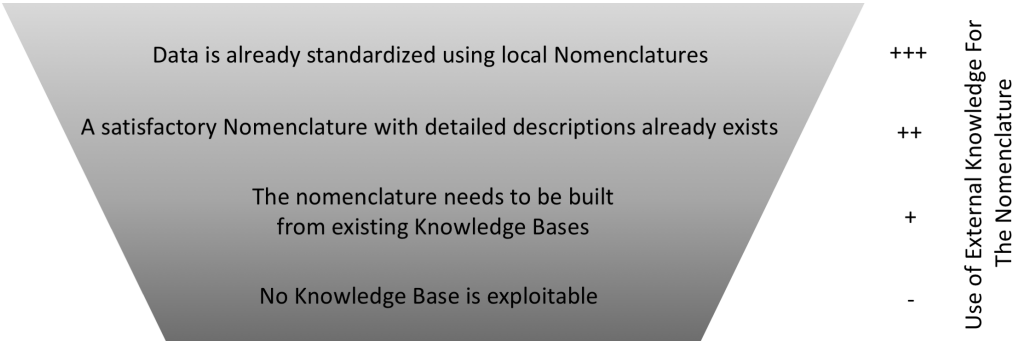


Figure 1.4 – Diagram illustrating the different levels of external knowledge that will be addressed in Chapters 3, 4, 5 and 6.

Standardization is discussed in this thesis from four different points of view, depending on the level of external knowledge, as illustrated in Figure 1.4. The formal definition of a document and of the standardization problem is given at the beginning of the related work (Chapter 2) and will be used in the rest of the thesis. Chapters 3, 4, 5 and 6 successively study standardization with a decreasing amount of external knowledge available:

- When the data has already been standardized. This means the text takes a value from a local nomenclature, namely the website's one. The fields *Contract*, *Study* and *Experience* are for instance manually selected by recruiters when posting the job. To leverage this rich knowledge, we proposed in [Malherbe et al., 2015b] a schema mapping solution based on case-based reasoning, which converts the local standardization into a common nomenclature.

- When a nomenclature with rich knowledge already exists. Such rich information is for instance expressed by semi-structured textual descriptions, as is the case for national *Job Categories*. The proposed semi-supervised standardization then aims to rank the categories using a learning-to-rank model [Malherbe et al., 2014], before enriching the categories using a corpus of job adverts as detailed in [Malherbe et al., 2015a]. When categorizing a job, the top-ranked category is selected, and a probability of correctness is

estimated using the supervised learning we proposed in [Malherbe et al., 2015c].

- When public sources appear to be suitable for building a knowledge base than can then be used as a nomenclature. These sources include the Semantic Web, Social Media and Professional Social Networks. As we successfully proposed in [Malherbe and Aufaure, 2016], the *Skills* and the *Educational Institutions* can be standardized using a nomenclature like this. The difficulty for the former is selecting the relevant entities and for the latter is performing data matching in order to deduplicate the sources.
- When it is impossible to build a nomenclature with rich knowledge. In this case, the only source of knowledge is a corpus of unannotated documents, for example the educational *Degrees* obtained by a candidate. In this case, we proposed in [Malherbe et al., 2016] an unsupervised approach based on a latent variable model, in order to generate a weakly labeled dataset, which will be the knowledge that will be used for standardization.

Chapter 7 gives the practical applications of the proposed standardization systems, that have all been implemented in an industrial context. The practical use of our work revealed some limitations, we will therefore propose possible future areas of research and work in Chapter 8.

Chapter 2

Related Work

Understanding the meaning behind unstructured text has been a major focus in research for decades. More recently, the data gained in size and heterogeneity with the explosion of the internet where the textual information is directly given by the users. Standardizing texts into domain specific nomenclatures is a process that can be done a posteriori, and gives the advantage of providing more exploitable data, both at the user level and at the computer level. It involves the following objectives:

- We need a nomenclature that represents the concepts in a comprehensive, generic and updated way. For every kind of textual attribute to be standardized, a different nomenclature is involved, and can be optionally associated to some meta-data.

- We need to associate every textual attribute to the corresponding concept. This process has to automatically follow the temporal evolution of the concepts, which happens when the nomenclature evolves but also when the vocabulary associated to the concepts evolves.

In this chapter, we will study the literature linked to these two aspects. The first one relates to the knowledge bases, which notably result from the Semantic Web initiative, and give information about entities. The fields of schema mapping, data matching and entity linking serve at manipulating and exploiting these knowledge bases, in a perspective of using them as nomenclatures. The second aspect relates to the machine learning, which aims to detect relevant patterns in a new documents, based on what has been seen on a data-set of documents. The supervised classification and the latent variable models serve in a sense at standardizing the documents into abstract nomenclatures, with however some limitations. In order to gain some clarity, the reader will be firstly introduced to the common representation and manipulation of textual data, that will be used in the whole thesis. This chapter will be concluded by the questions that appear to be crucial for standardization but remain unresolved in the literature.

2.1 Preliminary Definitions: Representing and Manipulating Texts

In order to clarify the notations and objects that will be manipulated in this chapter and the followings, we describe first the general approaches for representing a text, and the similarity measures that result. Table 2.1 summarizes the notations that will be used for the rest of the manuscrit.

2.1.1 Representation of Texts

A first step before manipulating a text is to give it a formal representation. Taken as is, a text is a chain of characters, and a first necessary process is to convert this into a list of processed words, and the common approach consists in the three following steps [Sebastiani, 2002]:

- Tokenization: The words are firstly separated, by splitting the text at each whitespace or punctuation character. This step is often coupled with an accents removal and a conversion of characters into lower case.
- Words filtering: A second step is to remove the words that do not convey any information. A word can be decided to be useless if it is too infrequent or too frequent in a given corpus, if it belongs to a certain grammatical category such as the determinants, or if it belongs to a list set beforehand called stopwords [Rajaraman et al., 2012].
- Words processing: This step serves at uniting some words having a very close meaning, for example "obey"/"obeying" and "car"/"cars", in order to reduce intelligently the set of words manipulated. To do so, a stemmer [Porter, 2001] can be used in order to keep only the stem of the words, or a lemmer [Liu et al., 2012] which gives the words lemmas.

After this process, the common assumption in text mining is to forget the order of the processed words [Lewis, 1998]. A text is then seen as a "bag of words" or *multiset* of terms $\boldsymbol{d}$, which is a set in which the same element can occur several times:

$$\boldsymbol{d} = \{\!\{t_1, t_2, ..., t_n\}\!\}$$

where each term t_i results from previous steps, and n is the number of such terms, varying on each text. This mathematical expression for a text opens the way to many calculations, like the similarity measures between two documents.

2.1.2 Similarity Measures between Texts

Leveraging the previous representation of texts can help to answer the following question: is the text $\boldsymbol{d}$ similar to the text $\boldsymbol{d}'$? For such computation, each term t of the text $\boldsymbol{d}$ is given a weight $w_{\boldsymbol{d},t} \in \mathbb{R}$. Many variants exist [et al., 1975], a key component of them being the term frequency $tf(\boldsymbol{d}, t) \in \mathbb{N}$ which counts how many times t appears in $\boldsymbol{d}$:

$$tf(\boldsymbol{d}, t) = |\{t_i \in \boldsymbol{d} | t_i = t\}|$$

In other words, the function $t \to tf(\boldsymbol{d}, t)$ is the multiplicity function of the multiset $\boldsymbol{d}$.

A second key component is the global weighting function [Berry and Browne, 2005], which does not depend on the text $\boldsymbol{d}$ but on a corpus of documents $\mathcal{D}$. It estimates the importance of the term t based on its appearance in $\mathcal{D}$, and is computed for all the terms appearing in the corpus, which form the vocabulary $\mathcal{V}$. For instance, the invert document frequency focuses on $1/p(t \in \boldsymbol{d} | \boldsymbol{d} \in \mathcal{D})$, which is the number of documents in the corpus divided by the number of documents containing the term t:

$$idf(t) = \frac{|\mathcal{D}|}{|\{\boldsymbol{d} \in \mathcal{D} | t \in \boldsymbol{d}\}|}$$

A variant of the previous function is used in the Okapi BM-25 [Robertson and Jones, 1976]:

$$idf_{BM25}(t) = \frac{|\mathcal{D}| - |\{\boldsymbol{d} \in \mathcal{D} | t \in \boldsymbol{d}\}| + 0.5}{|\{\boldsymbol{d} \in \mathcal{D} | t \in \boldsymbol{d}\}| + 0.5}$$

Closer to the information theory, the term entropy [Hotho et al., 2005] focuses on the probability that a document is $\boldsymbol{d}$ given that it contains the term t, written $p(\boldsymbol{d} | t \in \boldsymbol{d}) = \frac{tf(\boldsymbol{d},t)}{\sum_{\boldsymbol{d}'} tf(\boldsymbol{d}',t)}$:

$$entropy(t) = \frac{1}{log(|\mathcal{D}|)} \sum_{\boldsymbol{d} \in \mathcal{D}} \frac{tf(\boldsymbol{d}, t)}{\sum_{\boldsymbol{d}'} tf(\boldsymbol{d}', t)} log\left(\frac{tf(\boldsymbol{d}, t)}{\sum_{\boldsymbol{d}'} tf(\boldsymbol{d}', t)}\right)$$

To get the expression of $w_{\boldsymbol{d},t}$, numerous alternatives exist for combining the term frequency and the global weighting function [Berry and Browne, 2005], and the tf-idf is widely used to characterize documents [Sugiyama et al., 2003], including in many industrial implementations [Smiley and Pugh, 2011]:

$$w_{\boldsymbol{d},t} = \sqrt{tf(\boldsymbol{d}, t)} \times (1 + idf\,(t)) \tag{2.1}$$

Its variant, the Okapi BM25, has recently shown better results [Claveau, 2012]:

$$w_{\boldsymbol{d},t} = \frac{tf(\boldsymbol{d}, t).(k+1)}{tf(\boldsymbol{d}, t) + k.\big(1 - b + b.|\boldsymbol{d}|/\langle|\boldsymbol{d}|\rangle\big)} \times idf_{BM25}(t) \tag{2.2}$$

Where k and b are free parameters and $\langle|\boldsymbol{d}|\rangle$ is the average size of the texts in $\mathcal{D}$. The log entropy has also been found to work well in practice for many data-sets[Landauer et al., 2013]:

$$w_{\boldsymbol{d},t} = log\,(1 + tf(\boldsymbol{d}, t)) \times entropy(t) \tag{2.3}$$

Those weights are extended to terms t not in $\boldsymbol{d}$ by $w_{\boldsymbol{d},t} = 0$. Using them, a text is generally represented as a vector where each component corresponds to a term of $\mathcal{V}$ [Salton et al., 1975]. In this thesis, with a slight abuse of notations,

this vector will be noted $\overrightarrow{w_{\boldsymbol{d}}} \in \mathbb{R}^{|\mathcal{V}|}$, whose k-th component is given by the weight of the k-th term of the vocabulary $\mathcal{V}$, that is to say $w_{\boldsymbol{d},t}$ when one writes t this term. We emphasize the fact that the k-th component of $\overrightarrow{w_{\boldsymbol{d}}}$ is not related to the indexation of terms in $\boldsymbol{d}$ (Equation (2.1.1)).

For a given weighting $w_{\boldsymbol{d},t}$, a similarity measure between the two texts $\boldsymbol{d}$ and $\boldsymbol{d}'$ can be computed. Many alternatives exist, the most commonly used being [Huang, 2008]:

- the cosine similarity:

$$cos(\boldsymbol{d}, \boldsymbol{d}') = \frac{\overrightarrow{w_{\boldsymbol{d}}}.\overrightarrow{w_{\boldsymbol{d}'}}}{\|\overrightarrow{w_{\boldsymbol{d}}}\| \ \|\overrightarrow{w_{\boldsymbol{d}'}}\|} = \frac{\sum_{t\in\mathcal{V}} w_{\boldsymbol{d},t}.w_{\boldsymbol{d}',t}}{\sqrt{\sum_{t\in\mathcal{V}} w_{\boldsymbol{d},t}^2}\sqrt{\sum_{t\in\mathcal{V}} w_{\boldsymbol{d}',t}^2}} \in [0,1] \qquad (2.4)$$

- the Euclidian distance:

$$dist(\boldsymbol{d}, \boldsymbol{d}') = \|\overrightarrow{w_{\boldsymbol{d}}} - \overrightarrow{w_{\boldsymbol{d}'}}\| = \sqrt{\sum_{d\in\mathcal{V}} |w_{\boldsymbol{d},t} - w_{\boldsymbol{d}',t}|^2}$$

- the Jaccard coefficient:

$$\begin{aligned} jacc(\boldsymbol{d}, \boldsymbol{d}') &= \frac{\overrightarrow{w_{\boldsymbol{d}}}.\overrightarrow{w_{\boldsymbol{d}'}}}{\|\overrightarrow{w_{\boldsymbol{d}}}\|^2 + \|\overrightarrow{w_{\boldsymbol{d}'}}\|^2 - \overrightarrow{w_{\boldsymbol{d}}}.\overrightarrow{w_{\boldsymbol{d}'}}} \\ &= \frac{\sum_{t\in\mathcal{V}} w_{\boldsymbol{d},t}.w_{\boldsymbol{d}',t}}{\sum_{t\in\mathcal{V}} w_{\boldsymbol{d},t}^2 + \sum_{t\in\mathcal{V}} w_{\boldsymbol{d}',t}^2 - \sum_{t\in\mathcal{V}} w_{\boldsymbol{d},t}.w_{\boldsymbol{d}',t}} \end{aligned}$$

One notes that these similarities have the disadvantage to be purely keywords-based, as well as the representation of a document as a multiset. One better approach could be to map a text into a nomenclature, which leads to a concept-based text representation and a conceptual cosine similarity [Tan et al., 1999]. This approach is what we call standardization.

Symbol	Description	Nature of the object
t	Term	List of characters
$\boldsymbol{d}$	Document	Multiset of terms
$\mathcal{D}$	Corpus	Set of documents
$\mathcal{V}$	Vocabulary	Set of terms
$w_{\boldsymbol{d},t}$	Weight of the term t for document $\boldsymbol{d}$	$\in \mathbb{R}$
$\vec{w}_{\boldsymbol{d}}$	Term-weights vector for $\boldsymbol{d}$	$\in \mathbb{R}^{\|\mathcal{V}\|}$
$cos(\boldsymbol{d}, \boldsymbol{d}')$	Cosine similarity between $\boldsymbol{d}$ and $\boldsymbol{d}'$	$\in [0,1]$

Table 2.1 – Notations.

2.1.3 The Three Reasonings of Bob

Following the formulas above, we can teach Bob how to assess whether two documents are similar, as represented visually in Figure 2.2. This first kind of

reasoning just requires Bob to apply specific rules, and does not involve any memorized pre-computation. To illustrate that Bob can adopt different kind of reasoning, he will be represented in one of the three following states:

1. The observation, on the left-hand side of Figure 2.1, when Bob simply obeys to some fixed rules that he has been told, like when computing the similarity between two texts.
2. The learning, in the middle of Figure 2.1, when Bob is looking for some relevant patterns in a set of documents examples. He then keeps in memory these patterns, which are often represented as a vector of weights.
3. The prediction, on the right-hand side of Figure 2.1, when Bob is finding patterns in a new document, based on what he has computed and memorized during the learning.

Figure 2.1 – The three reasonings of Bob: the observation, the learning and the prediction.

As stated in Section 1.4, the different strategies in this thesis will be illustrated by our new friend Bob, represented in one of the states above. Now that the reader is fully prepared to read the illustrations with Bob, we can go deeper into the standardization problem, whose formalized statement is given in the following.

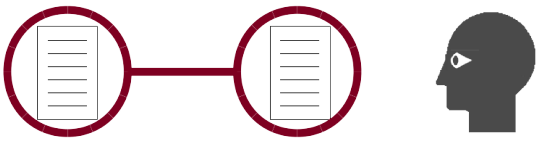


Figure 2.2 – Given a weighting $w_{\boldsymbol{d},t}$ and a similarity measure, Bob evaluates whether two documents are similar or not.

2.1.4 Formal Standardization Definition and Strategies

Before going further into the literature related to this thesis, we propose to express formally the standardization of the document $\boldsymbol{d}$. The problem is to define a function f such that:

$$f(\boldsymbol{d}) = c \in \mathcal{N}$$

where $\mathcal{N}$ is the target nomenclature that takes a finite number of values, and c is the concept associated to the document $\boldsymbol{d}$. The concept c is the standardized attribute resulting from standardization. To give example of $\mathcal{N}$, let us consider the list of the multinational companies, which would include a value n representing "SAP", with possibly meta information about this company: this is a nomenclature of entities. An example of abstract nomenclature $\mathcal{N}$ is the job categories.

A first interrogation is the nature of the nomenclature $\mathcal{N}$. The concepts it should represent is generally guided by industrial constraints, and can be limited by technical difficulties: a fine-grained nomenclature is often preferable, but might be replaced by a coarser nomenclature for which standardization is more feasible. Moreover, $\mathcal{N}$ can be the fruit of an automatic generation using external sources, mainly the internet. With this objective in mind the knowledge bases from the semantic web will be detailed in Section 2.2.1, and the matching of such sources is possible through the data matching presented in Section 2.2.2.

The second interrogation is the computation involved in f. In the case of a nomenclatures of entities, a natural approach is the entity linking detailed in 2.2.3, which involves a simple straightforward comparison following the remark in Section 1.3 that an entity is detectable only if it is explicit in $\boldsymbol{d}$. In the case of an abstract nomenclature $\mathcal{N}$, the classification literature detailed in Section 2.3.1 proposes to learn how to compute f from a data-set of examples $(\boldsymbol{d}, c)$, where the target concepts c are referred as classes. Last, in the section 2.3.2 are detailed the latent variable models for texts. Such models only leverage a data-set of documents $\boldsymbol{d}$ in order to compute a function f that projects a document into a cluster or some topics, which are not fixed in advance.

Section 2.2 and 2.3 address the literature related to the two paragraphs above. The former presents the existing nomenclatures $\mathcal{N}$ and their manipulation, and the latter presents possible computations for f, through supervised or unsupervised learning.

2.2 Manipulating Existing Nomenclatures

Determining the nomenclature to use is the very first step for standardizing documents. Building one from scratch involves high engineering costs and might not produce a generic nomenclature, so it is preferable to leverage some external source of information. The rapidly growing semantic web is a relevant source, it provides meta-data that enables to perform entity linking, but also raises the issue of multiple nomenclature, which therefore needs to be matched.

2.2.1 Knowledge Bases from the Semantic Web

The internet is an incredibly rich source of textual data, but without any processing it is difficultly exploitable. A major focus in the past decade has been to express the information in a computationally exploitable way following the recommendations given by [Berners-Lee et al., 2001]. The core idea is that the internet texts deal with concepts, for example "Company", "Google", "Search en-

gine", "Larry Page", each of which should be universally referred using a unique Uniform Resource Identifier (URI). A second core aspect to describe information is the Resource Description Framework (RDF) [Decker et al., 2000], which proposes to express predicates in form (subject, verb, object), for example ("Larry Page", "works-at", "Google"), which universally encapsulates the information that Larry Page works at the firm Google. In such description of the information, it is important to distinguish between the entities and the other concepts. Indeed, the entities are concepts that *exist*, such as persons and organizations, for example "Larry Page" and "Google". On the other side, there are the types, properties and interrelationships of the entities, for instance "company" is the type of "Google" and "works-at" is the relation between an employee and its company. It is critical that this second kind of concepts are carefully designed, because they structure the entities and their connexions - taken together, they form what we call an ontology. In an ideal world, there would be a unique ontology shared by everyone, but in practice, there exist numerous of them [Bizer et al., 2009a]. The project to express most of the internet information in a RDF structure is called the Semantic Web.

During the past decade, several open data projects have emerged with the purpose of building a knowledge base with information expressed in a RDF structure, leveraging one or several ontologies. For instance, WordNet [Fellbaum, 2010] aims to model the natural language. In this database the entities are words, and the ontology expresses relations such as hyponymy, hyperonymy or synonymy. This project does not deal with people, organizations or institutions, which is on the contrary a major aspect of the DBPedia project [Bizer et al., 2009b]. This other popular database is indeed built automatically using Wikipedia[1] pages, and entities are as various as Wikipedia pages, with celebrities, companies, movies or even softwares. Both of these projects have an encyclopedic purpose, and are not specific with a domain. Because of this, they use upper ontologies - to be contrasted with domain ontologies which are more specific.

When dealing with a domain, it can be very efficient to use a RDF-structured knowledge base. In e-recruitment, some ontologies have been proposed, in which the concepts were typically skills, degrees, institutions, companies and candidates. The real-world experiments or fictive scenarios gave promising results for matching CV with job adverts [Mochol et al., 2004, Fazel-Zarandi and Fox, 2009, Popescu and Popescu, 2010, Bourse et al., 2002], but the data have to be manually structured to fit the ontology: one has to specify the skills of the candidate, the degree level, and so on. When dealing with profiles or job adverts extracted from the internet, the attributes are not or poorly structured and an ontology-based reasoning is impossible. For concepts such as job sectors, job categories and skills, there exists national knowledge bases, for example ISCO [Winterton et al., 2009], O*Net [Hilton et al., 2010], and ROME [ROME, 2013]. Each of them can be seen as an attempt to build an universal structured database for e-recruitment. They are made up of entities which can be used as standards and enable to compute national job market statistics [Le Ru, 2015, Alterman

[1] urlhttps://www.wikipedia.org/

et al., 2008], but this requires to have structured manually the jobs and candidate profiles. For the problem of standardization, those structured databases are a good source of information, but are limited since they are not comprehensive and do not deal with concepts like companies or degrees. However, using directly these knowledge bases for analyzing the job adverts or candidate profiles correspond to a *top-down* approach. Indeed, these bases are predefined by recruitment experts. Consequently, they express a terminology and a knowledge that is not necessarily shared by the candidates nor the regular recruiters.

These knowledge bases represent rich structured information, and in this thesis we will use their concepts for our standardization nomenclatures. In light of this, we will not directly use the RDF-structured data, apart from extracting entities along with their associated meta information. In other words, the the knowledge bases can simply be seen in this thesis as *flat data-sets of entities* with their meta data associated. To the best of our knowledge, building a custom nomenclature is poorly tackled by the literature, since most systems simply rely on the existing knowledge bases as nomenclatures. Depending on the concept, some knowledge bases from the internet could be suitable sources for automatically building a nomenclature of entities. However, building a nomenclature from multiple sources implies to match the data, in order to avoid having duplicated entities in the final base.

2.2.2 Matching Hierarchized and Flat Nomenclatures

To represent the data in a unique way, one often needs to match two nomenclatures, which is done by matching every concept successively. When the nomenclature is short and can be hierarchized, the problem relates to schema mapping, whereas when it is large the problem is flat and relates then to as data matching.

Mapping Schemas of Concepts

A schema is a hierarchy or taxonomy of concepts, which can be the schema of a database, the XML structure of a webpage, or more specifically the nomenclature used in a recruitment website for the contract type or the required experience. When dealing with two websites or databases, the problem on interoperability can be solved by aligning the first schema on the second one, which means translating each term of the first into a term of the second, as represented in Figure 2.3. This problem is referred as schema mapping and has received a lot of attention over the past decade, with the growth of Internet and database integration [Shvaiko and Euzenat, 2013a]. The problem tackles the structural heterogeneity and semantic heterogeneity, because entities can be named differently and hierarchized differently. In the case of jobs, [Carlson and Schafer, 2008] proposes an automatic alignment of job adverts XML structures, which is equivalent to align the names of the attributes of Section 1.3. The schema mapping appeared to be more difficult for the nomenclatures of the attributes, and [Dorn and Naz, 2007] leaves this question open for job categories.

The schema matching task is still mainly done manually, with the potential use of crowdsourcing [Zhang et al., 2013, Viet et al., 2013], but the past decade

the manual efforts have been reduced thanks to semi-automatic mapping systems [Bellahsene et al., 2011]. There mainly exist three approaches to partially automatize the schema mapping:

- The instance-level mapping, which focuses on the instances associated to each concept, which is typically applicable when we match the schemas of databases
- The schema-level mapping, which focuses on the data type constraints associated to the concepts, the meta-data like the descriptions of the concepts and the structure of the schema which imply the concepts' names.
- The mapping-reuse, which leverages previously mappings performed to perform a new schema mapping.

Because of the multiplicity of the approaches, many systems are hybrid and associate some different matchers. Each matcher is independent and the combination of them is done a posteriori, using possibly machine learning [Rodrigues et al., 2015]. However in some cases, the only additional information we have about concepts are the name of the entities, and the structure; the two first approaches are then irrelevant. [Shvaiko and Euzenat, 2013b, Madhavan et al., 2005a] propose name-based approaches and the former computes the similarity between n-grams of words while the latter computes the Levenshtein distance between the names. Such similarities are direct and assume that names are the same for both schemas, meaning it will not work for two strictly different words with same meaning. The use of WordNet for extended semantic similarity seems to give good results, as proposed in [Manakanatas and Plexousakis, 2006] who separates names into tokens and leverage WordNet similarity between them. Such an approach highly depends on the choice of lexical database, and it would be preferable to leverage a domain specific base, containing for instance some terminology related to the e-recruitment, while WordNet appears to be too generic.

The mapping reuse is still not very spread out, although a corpus of manually matched schemas is often available because the industrial context required to match them. Among common match tools [Bernstein et al., 2011], only one uses this approach: named COMA, the system links schemas thanks to a golden standard schema, using transitivity of the semantic equivalence [Do and Rahm, 2002, Massmann et al., 2011]. Such a golden standard approach is time consuming, and can result in a loss of quality depending on the choice of the golden standard. [Manakanatas and Plexousakis, 2006] also reuses already identified mapping, but only when names to be mapped are exactly the same as for previous mapping, which is very restrictive. [Madhavan et al., 2005b] successfully proposes to use machine learning in order to learn a flexible similarity between names. To do so, they associate each concept with two sets of positive and negative matches, and vectorize the name, the description or instances of the concepts of those sets. This approach however suffer from a cold start problem and can only work when the source schema has already be mapped a lot, which is drastically restrictive in practice.

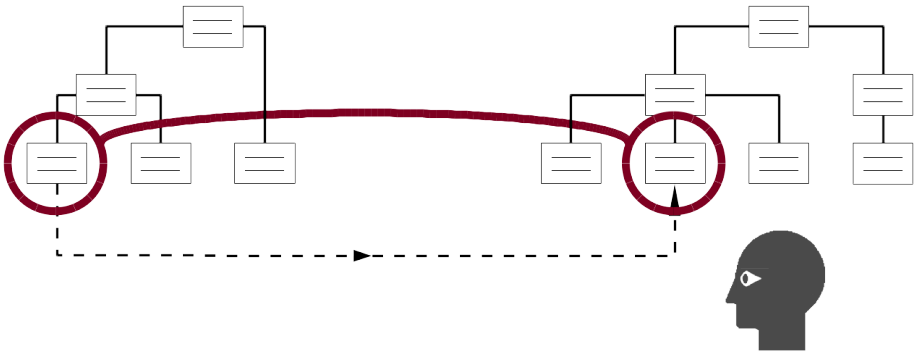


Figure 2.3 – Illustration of Bob's reasoning for the schema mapping: he observes a relation between the source item and its equivalent in the target schema. When it is a textual similarity that is observed, this corresponds to name-based mapping, but the observation can also involve for instance meta-data or data type constraint of the concepts.

In schema mapping, the match between concepts generally expresses an equivalence and not an equality, because every concept needs to be matched for interoperability and there is no insurance that an exact match exists. This applies for short nomenclatures, but with very long ones, typically with thousands of concepts, it can generally be sufficient to match the exact duplicates between the two sources, which is a task referred as data mapping.

Data Matching

When the data is not hierarchized and represents a list of entities, the data matching aims to find if the entities of the first source have an equivalent in the second source, or none. The task is then to merge duplicate entries from 2 data-sets, in order to create a unique deduplicated base. The domain is often referred as record linkage, because a recurrent use has been for merging data from different censuses [Herzog et al., 2010], but the typical applications covers data-sets of farms [Lavallée and Caron, 2001], restaurants [Ravikumar and Cohen, 2004] or e-commerce items [Christen, 2012]. Closer to our domain, data matching has been found to have critical application in deduplication of social network users [Buccafurri et al., 2012], but always considers the names alone and do not aim to deduplicate the attributes of the users. About universities, [Aumueller and Rahm, 2009] has applied data matching to the researchers' affiliations, which are regularly the laboratories themselves, contrary to a CV where any kind of institution can be found. All the previous examples involve entities, and no high-level concepts: while the schema matching can involve abstract nomenclatures, for example the industrial sectors, the data matching only applies to nomenclatures of entities, moreover fine-grained ones. In light of this, in data matching, the entries of the two sets are always directly comparable, because an entity systematically has one or few explicit names.

The problem has been formalized by [Fellegi and Sunter, 1969] as considering two sets A and B, and deciding for each pair $a, b \in A \times B$ whether to merge a and b or not, as illustrated in Figure 2.4. A and B can possibly be the same set, if the goal is to deduplicate the entities of a single data-set. The comparison of a and b is generally represented by a vector $\gamma(a, b)$, in which each dimension corresponds to the agreement or not regarding a feature, for instance the name or the address. The merging decision is then based on $\gamma(a, b)$ and can generally obey to deterministic rules or to a strategy which depends on conditional probabilities $p(\gamma(a, b)|a, b$ to be merged), directly computed from the data or estimated through the expectation maximization algorithm [Herzog et al., 2010]. Clustering or supervised learning on the vectors $\gamma(a, b)$ has also been successfully proposed by [Elfeky et al., 2002, Ravikumar and Cohen, 2004].

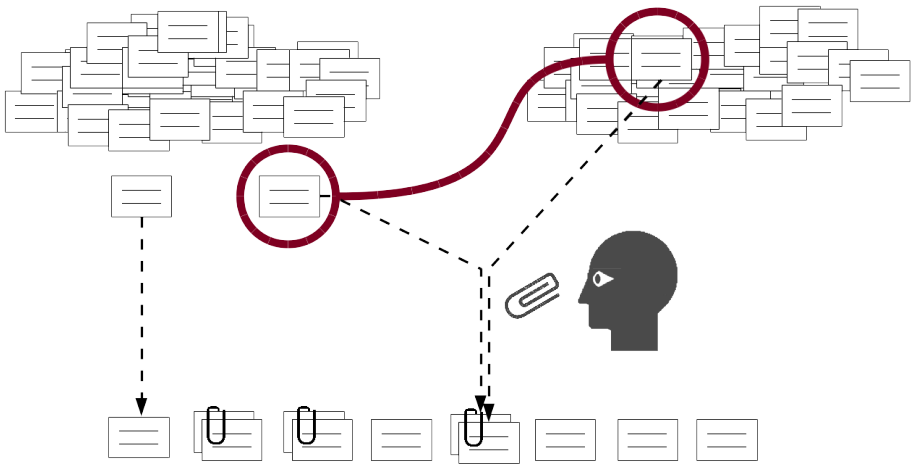


Figure 2.4 – Illustration of Bob's reasoning for the data matching: he staples the documents that are similar, in order to form a deduplicated stack of sheets of paper from the two original stacks.

As shown in Figure 2.4, from Bob's point of view, he is confronted to two sets of sheets of paper. The task is to staple the pairs taken from the two sets that represent the same entity. The approach is to directly compare the sheets of paper, comparison embodied by the vector $\gamma(a, b)$.

All methods except the supervised have concentrated a lot of efforts in the past decades, especially because no labeled data is required to learn the parameters. But without external information, we have no assurance that any of those methods provides a good performance. Indeed, all the metrics for data matching quality [Christen and Goiser, 2007] need to be computed on a labeled data-set. According to [Christen, 2012], majority of data matching techniques has unfortunately only been evaluated on a small quantity of labeled data, and in [Gomatam et al., 2002], no method clearly stands out, which both confirm the necessity for a proper evaluation on a consequent labeled data-set.

Nevertheless, constituting a labeled data-set for data-matching requires high human costs, and as [Winkler, 2006] recently pointed out, the manual review

process is not well documented. In particular, as made explicit by [Christen and Goiser, 2007], when considering all pairs $a, b \in A \times B$ the number of true negatives is extremely high, because many pairs do not need to be merged: this makes several classical metrics less meaningful, and a straightforward manual review strategy highly inefficient. Another important outcome having a labeled dataset would be to train models on the vectors $\gamma(a, b)$. In [Ravikumar and Cohen, 2004, Elfeky et al., 2002], heuristic approaches are compared with a supervised algorithm, which outperforms the other approaches. Both papers point out the difficulty of having labeled data, and in the experiments of [Elfeky et al., 2002], the training is done only on artificial data.

For the hierarchized and the flat data matching, the information about entities is used at matching the equivalent entities. This information is also used in entity linking, which aims to associate a mention to one entity of the nomenclature. This mention is unstructured, while until now the comparisons were typically between two structured values of nomenclatures.

2.2.3 Linking Documents to Entities of Nomenclatures

Given a knowledge base of entities, finding the entry that corresponds to a mention in a document is the problem of entity linking, a typical example being to link the company name of a job advert with the proper entity of a nomenclature of companies. The research about entity linking began relatively recently when [Cucerzan, 2007] and [Bunescu and Pasca, 2006] proposed to solve the disambiguation problem by linking given the mention extracted in documents to a Wikipedia page. The task gained into popularity with the challenge in the Text Analysis Conference TAC [McNamee and Dang, 2009], and is often used for named entity disambiguation, knowledge base population or textual data standardization. The entity linking process is generally separated into 3 steps [Hachey et al., 2013]:

- Extraction: Given the document and the mention, a query representing the problem is extracted. The query is often made up of the mention itself and some information of the surrounding context.
- Search: Based on the query, a set of relevant entities are selected in the knowledge base.
- Disambiguation: In the case where several candidate entities are selected, this step decides which one represents the mention.

The disambiguation step focuses on estimating the similarity between the query and each selected entity, that are represented using textual information contained in the knowledge base. From Bob's point of view, each box corresponds to an entity, and there are many boxes - as many as in the knowledge base. The entity's textual information is directly compared to the document that Bob needs to store, as shown in Figure 2.5. Indeed, the extracted query and the entity textual information are both the name of an entity, and are therefore directly comparable.

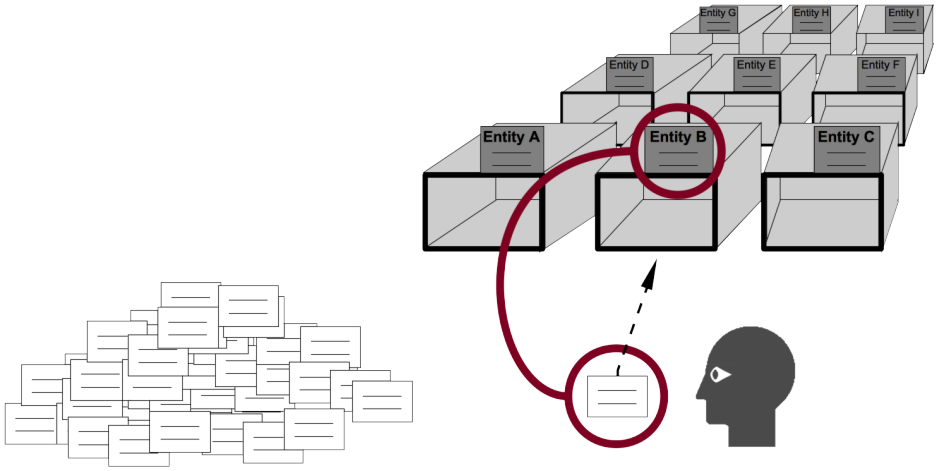


Figure 2.5 – Illustration of Bob's reasoning for the entity linking: Each box is taged with the name(s) of its entity. Bob places the sheet of paper into the box whose tag is the most similar to the sheet of paper itself.

The system can also return the answer NIL, when no entity seems to represent the mention after the search or disambiguation steps. A vast majority of the proposed systems in the literature leverages the information of wikipedia for the entities features [Rao et al., 2013], using for instance URL redirections and disambiguation pages as aliases for the entity. Such data is made easily accessible through the DBpedia project previously described in Section 2.2.1, and is considered as a sufficient source of knowledge in most of the papers. Consequently, the literature does not treat the creation of the knowledge base, and the common approach to face queries that have no corresponding entity is to return NIL, in order to potentially add it manually in the base. Some work designated as knowledge base population aims to enrich existing entities in the knowledge base [Ji and Grishman, 2011], but a real need is to create totally new entities by considering for example multiple knowledge bases.

Similarly to data matching, designing an entity linking system involves evaluating it, which requires a labeled data-set. The labeled data also find a significant importance when supervised learning is used for linking, which have been found to be a performant approach [Zhang et al., 2010, Rao et al., 2013]. The usual entity linking metrics [Hachey et al., 2013] endeavor to analyse separately the accuracies of the search step and the disambiguation step. This enables to tune efficiently the system on each step separately. In the case of a knowledge base built up from multiple sources, evaluating the entity linking would imply to define adapted metrics.

The task of entity linking is only possible when there exists a nomenclature of entities, with information for each of them. However, when we want to link a document to an abstract nomenclature, for example the job categories, we can not have some descriptions of the concepts directly comparable with the document to standardize. In practice there might not be any description of the

concepts. In such case, we need to focus on the document to standardize and find patterns in it.

2.3 Standardizing by Finding Patterns in Documents

In order to learn relevant properties in textual documents, we can leverage a set of documents which will be our guiding examples. These documents might have been manually labeled: in this case, the labels are referred as classes, and finding the patterns in each class is referred as supervised learning, that will be the first part of this section. On the contrary, when the documents have not been manually labeled, the learning is unsupervised and aims generally to determine clusters of documents or finding the topics in them. A major trend for such analysis are the latent variable models, that will be studied in the second part of this section. In each case, the set of topics, of clusters or of classes can be viewed as an abstract nomenclature, since they correspond to higher level concepts and not to existing entities.

2.3.1 Learning to Classify from Examples

A big range of problems can efficiently be solved by leveraging a set of manually solved examples; this is in particular true for many standardization problems. This approach is called the supervised machine learning, and the examples are called the training samples. The problem of classification is a kind of supervised learning in which the objective is to give a class to each input data. If the input data is textual documents, the examples can be represented as a data-set of classified documents $(\boldsymbol{d}, c)$ where the class c is a concept of an abstract nomenclature, for example the job category of a job description $\boldsymbol{d}$. As shown in Figure 2.6 with the metaphor of boxes that represent classes, the process separates into two steps: the training, during which we learn the patterns associated to each class by examining the labeled documents; the prediction, when the label of a new input document is inferred by examining which pattern is encountered in it. We describe here three ranges of classifiers popular for text mining [Sebastiani, 2002], the proximity based classifiers, the linear classifiers, and probabilistic classifiers.

The most easily understandable classifiers are based on the proximity between documents, and have thus a *geometrical* interpretation [Gower and Ross, 1998]. For documents, it requires to represent documents as vectors $\overrightarrow{w_{\boldsymbol{d}}}$ [et al., 1975] where each dimension corresponds to a term of the vocabulary $t \in \mathcal{V}$, and the value is the weight $w_{\boldsymbol{d},t}$ where Equations (2.1), (2.2) and (2.3) give some of the most popular variants (see Section 2.1). One historic classifier is the nearest neighbors [Cover and Hart, 1967], whose idea is to find the example of the training set which is the most similar to the input document $\boldsymbol{d}$, and consider as solution the class of this example. The training step does not involve any computation but the prediction involves to compare the document with each sample, so that its complexity increases linearly with the size of the training set. The Rocchio classification [Rocchio, 1971] scales better to large training sets. It proposes to compute the average vector of each class, also called centroid, and prediction

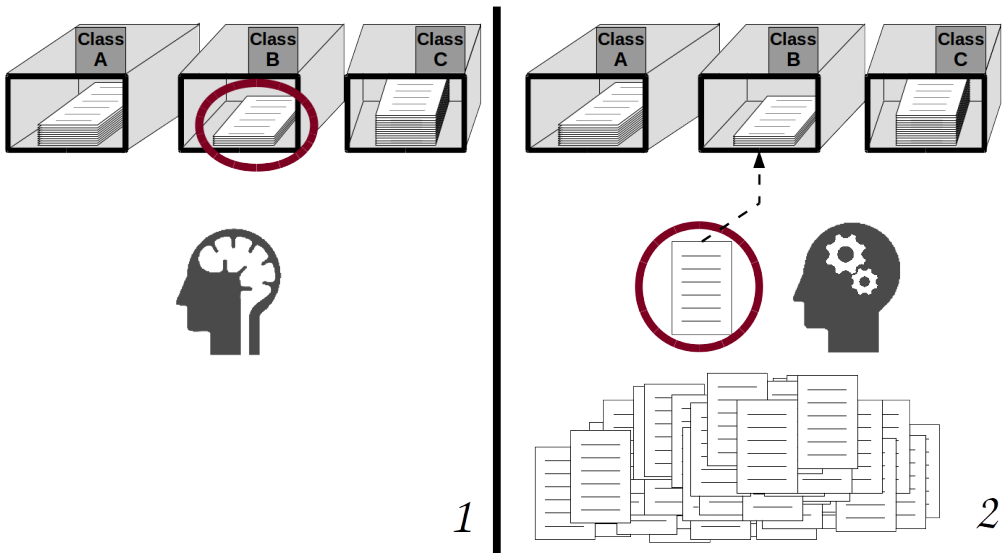


Figure 2.6 – Illustration of Bob's reasoning for the classification: 1. For each box, Bob learns what are the typical patterns of the documents inside. 2. Bob predicts a box for a new document based on the patterns he found in it.

is then limited to comparing the input document with each classes' centroid. Both classifications are fairly simple, but suffer from overfitting, meaning they are efficient when the input document is very close to the training samples, but can be highly wrong for different documents from what was seen in the examples.

A second group are binary linear classifiers, who split the vector space into 2 regions separated by an hyperplane. Each region modelizes a class, and the two classes are usually a positive and a negative one. The parameters to be learned are the parameters of the hyperplan, which are the normal vector to the hyperplane $\vec{\beta} \in \mathbb{R}^{|\mathcal{V}|}$, and the offset $\alpha \in \mathbb{R}$. One example is the Logistic Regression, for whose probability for a vector $\vec{x}$ of being in the positive class is modelled by $\frac{1}{1+e^{-\vec{\beta}.\vec{x}+\alpha}}$. It can be trained using gradient descent, and is a particular case of the perceptron, i.e. the single layer neural network. A more popular algorithm is the Support Vector Machines. The interpretation is purely geometric, it aims to find the separating hyperplane that is the most distant from the samples of each class. The training is the optimisation of the function $max_{\vec{\beta},\alpha}\|\vec{\beta}\|$, under constraints that express the separation of the samples. It can be extended to other kernels, but the linear assumption gives good results for text classification [Andrews et al., 2002]. The linear classifiers generalize better than Rocchio and nearest neighbors classifiers, but the number of parameters to learn is very high, since it equals the size of vocabulary $|\mathcal{V}|$: a very high number of labeled samples are required for the training. Furthermore, the binary classifier are originally defined for bi-class problems, and a multi-classes problem is solved by training a classifier for each class before combining them: this means that we need for each single class a high number of labeled samples.

Another range of classifiers are the Naive Bayes, that are probabilistic, be-

cause we compute $p(\boldsymbol{d}|class\ c)$ from a model in order to infer $p(class\ c|\boldsymbol{d})$. The models are *generative*, meaning we consider that a class "gives birth" to a document, and the computation of $p(\boldsymbol{d}|class\ c)$ does not require to represent the texts in the vector space but only as multisets of terms (Equation (2.1.1) of Section 2.1.1). The two popular models for texts are the Bernoulli and the Multinomial Naive Bayes. The first one assumes that each term of the vocabulary $\mathcal{V}$ can be present or not in the document $\boldsymbol{d}$: we generate the document by determining for each term $t \in \mathcal{V}$ if the event $t \in \boldsymbol{d}$ is true or false. We compute these probabilities $p(t \in \boldsymbol{d}|class\ c)$ for all terms $t \in \mathcal{V}$ and multiply them to get $p(\boldsymbol{d}|class\ c)$, as shown in the left-hand side of Figure 2.7. The Bernoulli model therefore penalizes the absence of every term not in $\boldsymbol{d}$, which is discussable especially when the vocabulary $\mathcal{V}$ is large. In a different way, the Multinomial model assumes that given a class, the document is generated term after term t_i, following the event $t = t_i$ for a number of n times, n being the number of terms in $\boldsymbol{d}$. The event does not depend on i but only on the distribution of terms per class, and we multiply the n values $p(t = t_i|class\ c)$ to get $p(\boldsymbol{d}|class\ c)$. In both cases, the training computes a formula corresponding to the maximum likelihood estimation. If it has a naturally a multiclass classification, it also requires to learn $|\mathcal{V}|$ parameters.

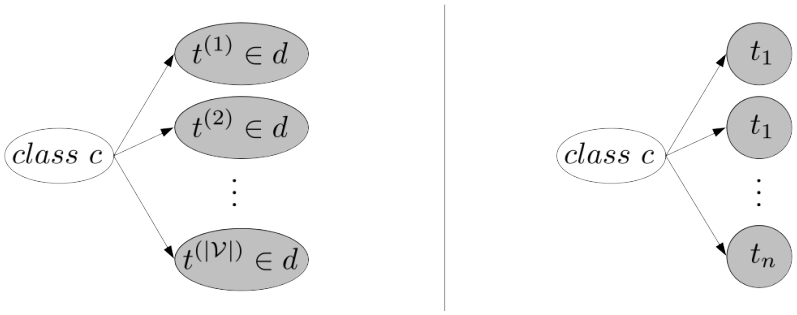


Figure 2.7 – Bayes diagram describing two Naive Bayes algorithms. On the left, the Bernoulli version generates the presence or not of each term of the vocabulary $\mathcal{V}$, noted $t^{(i)}$ for $i = 1..|\mathcal{V}|$. On the right, the Multinomial version generates successively the n terms of the document $\boldsymbol{d}$.

The previous models gives good results for textual classification [Wang and Manning, 2012], and can solve a standardization problem provided that we have a sufficient number of documents in the training set. Indeed, the dimension of the problem, namely the number of words in $\mathcal{V}$, is of the order of minimum $10,000$. Consequently, the number of documents has to be of the same order, and every class must be significantly represented, which is problematic in the case of fine grained classification. One way to train a classifier on fewer documents is to use dimensionality reduction, which aims to represent documents in fewer dimension using a preprocessing unsupervised learning. Along with the latent variable models described in the next section, the most popular ways to reduce a problem's dimension are the principal component analysis [Jolliffe, 2002] and the latent semantic analysis [Dumais, 2004], which both are algebraical algorithms that correspond to the diagonalization of a matrix. However, even after such

pre-process, the classification precision might still be low; in this case, a strategy could be to require additional manual validation.

The active learning proposes to efficiently leverage the manual validation, by interactively querying a user about the class of some input data. The objective is to improve incrementally the classifier, by choosing optimally the input data to be manually label by the user [Settles, 2010]. The literature has widely focused on the approaches for choosing what data should be manually labeled, and the most simple and used approach is to consider the most uncertain classifications, meaning those with the lowest probability $p(c|\boldsymbol{d})$ in the case of a document $\boldsymbol{d}$ with a predicted class c. For fine-grained classification of documents, following the previous paragraph, we would need a very high number of manual validations to significantly improve the classifier.

Instead of focusing on improving the classifier - which is the initial objective of active learning - one might instead consider that the manual validation is an integrant part of the system, and similarly focus on the cases' uncertainty. In other words, we can consider that some human labour will always be necessary for classifying documents $\boldsymbol{d}$, but we can limit the validation to only uncertain cases. This means we need to automatically detect the uncertainty of classification given by the probability $p(c|\boldsymbol{d})$. This quantity is naturally produced by probabilistic classifiers, but not by others classifiers. Any classifier outputs a raw score, that can give a first insight of uncertainty, but it can be more relevant to estimate a probability instead. Indeed, the probabilities are normalized scores, and they provide better user experience, especially for non expert users [Kruppa et al., 2014b]. In case of a binary SVM, the raw score is an non-bounded distance to the separating hyperplane, and the probability can be accurately estimated by fitting a logistic regression on the distances [Platt, 1999]. The logistic regression raw scores can indeed be interpreted as probabilities [Hosmer Jr and Lemeshow, 2004] and are widely used in probability estimation. Recently, it has been shown that multi-dimensional logistic regression enables to estimate the probability for the K-NN and Rochio classifiers [Kruppa et al., 2014a], which both produce naturally multi-dimensional scores. Furthermore, a combination of logistic regressions can also provide easily a probability interpretation while having better performances. In [Nie and Fei, 2014], the combination is performed through an Adaboost, meaning the classifier is a weighted sum of several logistic regressions, and the probability is estimated using a different formula that avoids too close values from 0 and 1. For a multi-class classification computed through one-versus-all binary classifiers, one can directly estimate independently the posterior probabilities for each classifier, by fitting independent logistic regressions. However an additional computation can significantly improve the probability estimation. Indeed, the approach proposed in [et al., 2004, Zadrozny and Elkan, 2002] combines the previous one-versus-all probabilities in order to refine the estimation incrementally. However, all these studies need a consequent training set, including significant number of positive cases for every class, which is limiting for fine grained classification of documents.

In all its previously described forms, the classification of documents has found many applications, including in e-recruitment for categorizing jobs. This problem

is a example of standardization, and has for instance been used for managing human ressource [Cornelius et al., 1979], for matching a job advert to a candidate resume [Diaby and Viennet, 2014], or more recently for predicting recruitment campaigns success, as proposed in [Séguela, 2012]. The general need is that a job advert has conceptually a category, for example "software engineer" or "sales manager", and that it can not be captured by some simple keywords, as pointed out by the design of jobs search engines [Schmidt et al., 2015]. One specific aspect of the categorization of job adverts is that the categories nomenclature is usually fine-grained, and in literature the number of classes is at minimum of one hundred [Séguéla, 2011, Séguela, 2012, Ghani, 2002, Schnitzer et al., 2014, Javed et al., 2014]. These nomenclatures are generally organized as a taxonomy, with a first level, coarser, counting several dozens of categories, and a second one, finer.

For instance, [Zhang et al., 2012] proposes for a Chinese recruitment website to disabled people to separate the job adverts into 7 employment sectors. They train a variant of the Naive Bayes classifier on a training set of 5,700 labeled job adverts, leading to a precision and a recall high enough to automatically categorize real-world job adverts, but remains on a coarse grained classification. In [Schnitzer et al., 2014] the job adverts can be associated to several categories out of 103, which is therefore a multi-label classification. In their dataset, the 10,300 adverts are on average associated with 4 categories. They propose to train several binary classifiers before combining them in an ensemble method, which gives excellent results when they restrict the classes to the number of 5. Moreover, when the learning is done on a limited dataset of only 1,000 job adverts, a manual validation is necessary for ambiguous cases. In [Séguéla, 2011, Séguela, 2012], a linear SVM is trained on a data-set of 9,300 job adverts. The results on the first level (23 classes) shows that the recall and precision really depend on the considered class: the job adverts in "Human Resources", "Accounting" and "Computer Science" seem to have a much more specific vocabulary, and are therefore better classified, contrary to the adverts in "project management" and "clients support". However, the data-set appears to be too limited to consider classification on the second level of 107 classes, and when they propose to reduce the dimension through PCA, even if they gets slightly better results with only 300 dimensions - compared to about 10,000 words - it remains unsatisfactory for fine-grained level classification. [Ghani, 2002] proposes to use ECOC [Hatami, 2012], which is particularly indicated for high number of classes, because it decomposes the m-classes problem into a number of binary problem lower than m, and scales up sub-linearly with the number of classes. This algorithm gets good results for classification on 65 categories, but the training required 132,000 labeled job adverts.

As the literature about job categorization demonstrates, the fully supervised classification appears to be complicated for fine-grained categories as it requires a significant labeled training set. Furthermore, in order to follow the evolution of the job market, we would need to update this data-set regularly. One option that still remains poorly tackled in literature is to leverage one of the existing nomenclatures. The latters generally provide rich information, as do the national databases discussed in Section 2.2.1. The only related approach [Javed et al.,

2014] studies the classification on the first level of the american national taxonomy for jobs O*Net [Dierdorff et al., 2013], and proposes to train a linear SVM on 3,6 millions of labeled job adverts. They vectorize only the job title, since it conveys more information than all other attributes of a job advert, according to [Diaby et al., 2013]. The experiments required a consequent infrastructure, 16 computing units with 35BG of RAM, and gave very promising results for 25 categories, but they leave the consideration of finer level as a future perspective. In particular they only use the structure of the O*Net nomenclature, and not any meta-data associated to it. Since the meta-data of a nomenclature might include the concepts' descriptions and titles and is regularly updated by the domain experts, it might be extremely relevant to use it for the classification.

However, for many standardization problems, the nomenclature is not associated to any meta-data, and we might have no labeled documents - or only a limited number. In such case, the only option is to leverage a data-set of unlabeled documents through unsupervised or semi-supervised learning, as detailed in the following.

2.3.2 Unsupervised Latent Variable Models for Documents

The unsupervised branch of machine learning leverages only unlabeled data as input and has found many application in text mining, one typical task being to cluster hundreds of thousands of job adverts into groups that share the same skills [Litecky et al., 2010]. One sub-class of unsupervised learning are latent variable models, which assume that each example is generated by an unknown latent variable. Such models are probabilistic and generative by definition. They can be easily illustrated by bayesian networks [Whittaker, 2009], which are graphs that represent the conditional dependencies between the random variables.

Historically, the first model of this kind is the mixture of gaussian, proposed by [Pearson, 1894]. He realized that the studied crabs sizes followed a first normal law for a subpart of them, and another normal law for others. In this example, the latent variable represented actually the two subspecies of the crabs. In order to generate the size of a crab, one has to flip a biased coin to determine the crab species, before sampling a size following the normal law of this species. The same idea has been applied to documents [Nigam et al., 2000], meaning that one assumes that the input documents form several groups in which the words have a specific distribution. The law for such distribution can be Multinomial or Bernoulli, which corresponds to a Naive Bayes Classifier but with the class of the document considered as a latent variable 2.7. To generate a document, one has to flip a biased coin to determine the class c, before sampling the words with respect to the class words distribution. The model's parameters are then computed using a Expectation Maximization algorithm (EM) [Moon, 1996], which is very popular for latent variable models [Everett, 2013]:

1. Initialization: for each input document, assign a value to the latent variable z. Without any specific clue, this assignment is fully random.

2. While the expected total log-likelihood $Q = \sum_{\boldsymbol{d}} \mathbb{E}_{z|\boldsymbol{d}}\Big(log\big(p(\boldsymbol{d})\big)\Big)$ has not

converged, repeat the following steps:

(a) Maximization (M-step): learn the parameters of the generative model $p(\boldsymbol{d}|z)$, considering the probabilities for the values of z of every document.

(b) Expectation (E-step): for each document $\boldsymbol{d}$, compute the new probability for each possible value of z, based on $p(z|\boldsymbol{d}) \propto p(\boldsymbol{d}|z)p(z)$. After this step the expected total log-likelihood value Q is updated.

At the end, one can be interested in the final values z assigned to each document, as well as in the model linking documents to classes $p(\boldsymbol{d}|z)$ and its parameters. This baseline model is also called mixtures of unigrams, and is at the origin of several models in the domain of semi-supervised learning, clustering and topic modeling [Blei, 2004].

The mixture of unigram was originally a semi-supervised algorithm [Nigam et al., 2000], meaning the solved problem is a classification where the training involves labeled documents but also *unlabeled* ones, in order to increase the precision compared to training on labeled documents only. To do so, the labeled data are used in the initialization for computing the probabilities $p(\boldsymbol{d}|class\ c)$, as well as in the M-step when learning the parameters on both pre-labeled and automatically labeled documents. The semi-supervised form of the mixtures of unigrams has been compared in the literature to another latent variable model, the co-training [Nigam and Ghani, 2000], which does not require an EM approach. The idea is to split cleverly each labeled document in two sub-documents $\boldsymbol{d} = (\boldsymbol{d^a}, \boldsymbol{d^b})$ and learn one classifier for the data-set with the sub-documents $\boldsymbol{d^a}$, and another one for the sub-documents $\boldsymbol{d^b}$. Both data-sets are incrementally increased with the unlabeled documents, by considering in priority the most certain predictions according to one classifier. This way, the unlabeled documents are progressively labeled while the classifiers are improved thanks to these new training examples. One assumption that provides theoretical guarantees for the co-training is that $\boldsymbol{d^b}$ does not depend on $\boldsymbol{d^a}$ given the document's class c [Blum and Mitchell, 1998], which is theoretically a latent variable model as shown in Figure 2.8. However the documents are not labeled using $p(c|\boldsymbol{d}) \propto p(\boldsymbol{d^a}|c) \times p(\boldsymbol{d^b}|c)$ but using only one of the two last probabilities - the highest one. In practice, the quantity representing the classifications' certainty is not necessarily a probability; it needs however to be precisely estimated, as for the active learning (the co-training can be seen as an automated active learning). The co-training applies to documents having an obvious split into 2 subspaces that each conveys rich information. This is the case for the job adverts since one can separate their title and their description [Ghani, 2002]. In this latter paper, they leverage the amount of unlabeled adverts available on the internet by using co-training with a simple Naive Bayes. This approach enables them to train an algorithm on only 13,200 categorized job adverts in a nomenclature of 65 classes, thanks to 120,000 unlabelled job adverts. They compare the co-training to the basic EM detailed previously and to a combination of both that they propose, called co-EM. They get the best results for the last one, with a precision on 65 categories equivalent to the supervised approaches on coarser categories [Séguéla, 2011], but the consequent size

of their labeled data-set of job adverts makes their system difficultly updatable for following the new trends of the job market.

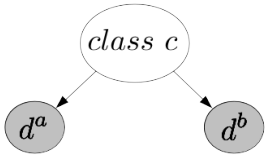


Figure 2.8 – In the co-training model, the class generates separately the two sub-documents $\boldsymbol{d^a}$ and $\boldsymbol{d^b}$. When applied to a job advert, $\boldsymbol{d^a}$ is the title and $\boldsymbol{d^b}$ is the description.

In the case where there is no labeled data at all but only a corpus of documents, the problem of clustering aims to form groups of documents. The meaning of those groups is unknown, contrary to the classes in the supervised case. As illustrated in Figure 2.9, from Bob's point of view, he is asked to group similar documents together, which then enables him to assign a cluster for new documents. Several clustering algorithms are geometrical, such as the K-Means in which each cluster's centroid is iteratively computed, but for analyzing textual data the probabilistic clustering has been more used [Aggarwal and Zhai, 2012], which corresponds to latent variable models. One of them that has showed great efficiency [Cornell, 2011] is the previously detailed model of the mixture of unigrams [Blei, 2004] with a random initialization and a M-step computed only on initially unlabeled documents; this model can be seen an unsupervised version of the Multinomial Naive Bayes in its fuzzy variant [Li and Deogun, 2009]. A point worthwhile to notice is that clusters have no interpretation, and to automatically give a meaningful name to each cluster is a difficult task [Tseng, 2010], which is therefore generally done manually a posteriori. Contrary to the classification, to assign a cluster number to a document can not be considered as a standardization, since the clusters do not represent meaningful concepts. They indeed just represent groups, and are therefore not a suitable nomenclature, apart if a highly costly manual review reveals they also represent meaningful concepts.

In literature, clustering have been a support of classification rather than a replacement. For instance, [Sasaki and Shinnou, 2005] proposes to cluster documents before classifying them, in order to capture their topic and improve the supervised spam detection, which is a kind of data reduction, whereas [McCallum, 1999] tackles the text classification by using mixtures that are trained for each class. The clustering has repeatedly found application in the e-recruitment, for example on the job adverts by [Javed et al., 2014] who plans to go beyond the O*Net taxonomy, by performing clustering on millions of jobs automatically categorized. Their work does not aim to improve the categorization but at finding new niches and emerging job titles, and relies on massive data-set of annotated job adverts and computing infrastructure. The clustering has also been used for analyzing candidate profiles, for instance by [Karamatli and Akyokus, 2010] who clusters the entities extracted in a resume based on the closeness of their positions, in order to group the education and work experience information, before

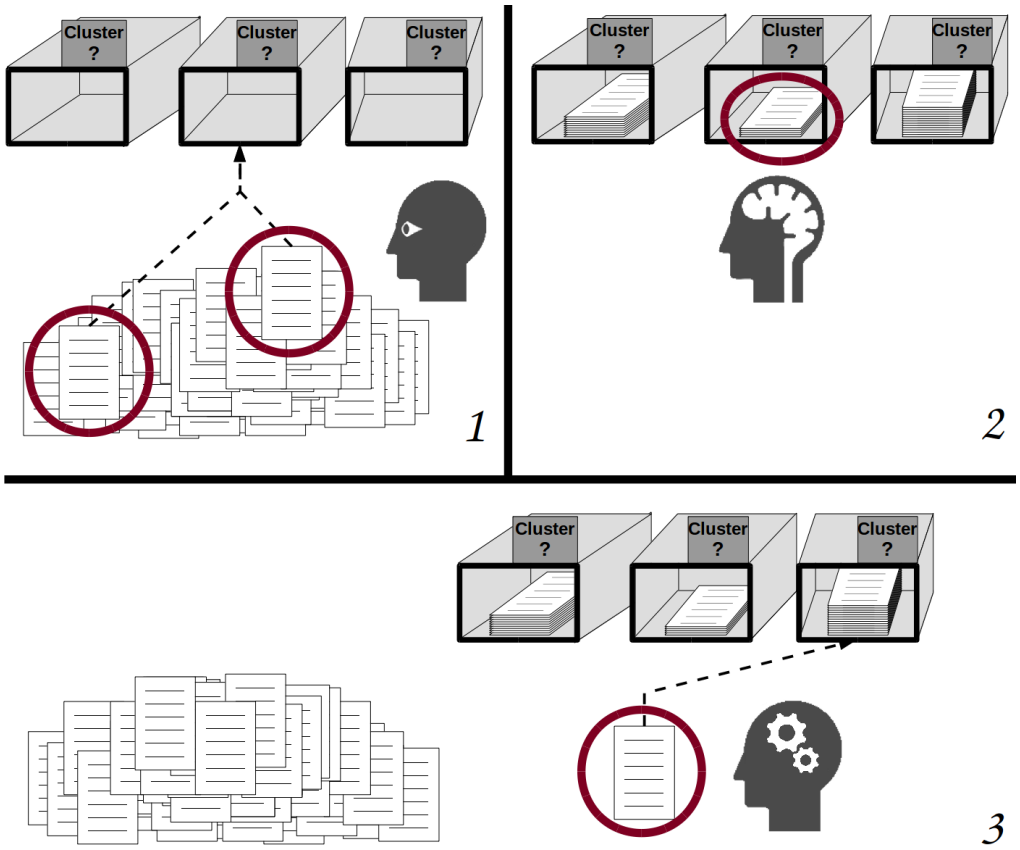


Figure 2.9 – Illustration of Bob's reasoning for the clustering: in the first frame, Bob groups the documents that are similar in a certain sense - strictly speaking, this is the clustering. In the second frame, Bob learns the properties of the clusters. In the third frame, Bob predicts the cluster of a new document. Each box is simply numbered, and has no meaningful name, since the cluster's meaning remains a mystery to Bob.

standardizing them using some dictionaries. [Hong et al., 2013] finds clusters of candidates of a recruitment website based on their textual information as well as their click and search frequencies, in order to use different job recommendation strategies for each cluster. They manually determine those strategies after having manually interpreted those 3 clusters. In [Sivaram and Ramar, 2010], the clustering aims to detect relevant candidates for a company, based on the textual information manually standardized. Following a comparison with a supervised classification, they conclude that K-means and its fuzzy variant are not applicable for resumes.

A more recent use of the latent variable models for text are the topic models [Aggarwal and Zhai, 2012], which are closely related to the probabilistic clustering presented above. The difference is that such models do not aim to form groups of documents but groups of terms, which represent the topics covered in a corpus of documents. A given document is then assigned to several topics supposed to capture the its global themes, instead of a unique cluster in the general case of clustering. The topics are unknown a priori, and could for instance capture the vocabularies related to the car industry, to software development or to the working conditions. A first topic modeling is the Latent Semantic Analysis or LSA [Deerwester et al., 1990], which is purely algebraic and not probabilistic, since it works by diagonalizing the matrix of words co-occurence in the input corpus. The idea is that the eigen vectors would regroup the words occurring in the same documents, which therefore represent the same topic. But rapidly this first deterministic approach has been supplanted by the Probabilistic Latent Semantic Analysis [Hofmann, 2001], which is a latent variable model in which the same word can be associated to different topics depending on the document where it appears. As shown in Figure 2.10, for a model with K topics, a document $\boldsymbol{d}$ of size n is generated in three steps: first, one latent variable of dimension K is generated for representing the distribution of topics of $\boldsymbol{d}$. Then, following this distribution, n latent variables $z_i \in \{1, ..., K\}$ are generated, and every term t_i is finally generated with respect to the terms distribution $p(t_i|z_i)$ for the topic numbered z_i. A popular evolution of this model has been proposed by [Blei et al., 2003] by adding the assumption that the distributions of topics per document as well as words per topic are Dirichlet distributions, meaning that a document only covers few topics and a topic is depicted by only few words. This model is the Latent Dirichlet Allocation or LDA and is considered as the most popular topic model [Blei, 2012, Arora et al., 2012].

Like other topic models, the LDA is hard to quantitatively evaluate; but it is known to provide fairly interpretable topics by humans. It has been successfully applied on corpuses of job adverts, for instance in [Kim et al., 2011] where they manually find the "abusive" LDA topics in order to automatically filter the job adverts of some freelancing sites. [Thompson and Willis, 2015] proposes to replace the qualitative analysis of job adverts by just giving the list of the topics' labels, and to do so they train a LDA with 200 topics on 15,000+ job adverts from Indeed, before manually labeling the computed topics. They compare the predicted topics with the cluster predicted using a K-mean with 50 clusters, and conclude that the LDA is more appropriate because there are several aspects

in a job advert that are not captured by the clusters, too coarse. Consistently with this statement, [Mimno and McCallum, 2008] finds the LDA topics to be very promising for predicting the sequence of job positions in a résumé. Their point is that 85 percent of job titles are unique out of the 54,000 positions they study in a corpus of about 10,000 resumes, which means job titles are too specific and useless for prediction. To cope with this, they propose to first compute 200 topics using the LDA, and then learn a generative sequence model in which each position is assigned to a professional state which corresponds to a mixture of topics. They evaluate the model by comparing the log-likelihood values of different variants, which poorly indicates the performance but points out that the model with states and topics is most efficient, and in particular compared with a model that uses just titles.

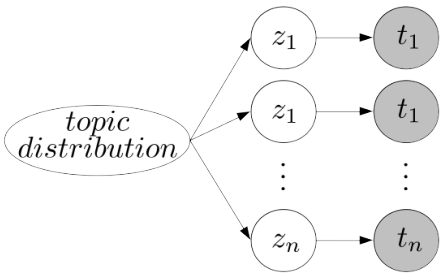


Figure 2.10 – Diagram describing the probabilistic LSA, in which the topic distribution generates latent topics z_i associated to each term t_i ($i = 1..n$ where n is the number of terms in $\mathbf{d}$).

Another interesting aspect about [Mimno and McCallum, 2008] is that topics are automatically given a label which is the most frequently associated job title in the corpus. This labeling leverages the meta-data provided by resumes, and an in-depth use of the meta-data would be to consider it in the computation of the topics, which is a global prospect of topic modeling [Blei, 2012]. In [Mimno and McCallum, 2012], and similarly in [Mei and Zhai, 2006], the distribution of topics depends on the document features, for example the author, the location or the date, each being given as meta-data. Some work about temporal text mining [Mei and Zhai, 2005] combines topic models with information about temporal evolution, but these two aspects are computed separately one after the other, as it is done in [Mimno and McCallum, 2008]. The latter work confirms that latent variable models can be relevant for sequence of documents, for example the work positions or educations in a candidate profile. However, their model do not consider the sequentiality for the computation of the topics. If meta-data could enable us to know what each topic or cluster represents, the topic modeling and clustering could then serve at standardizing documents.

2.4 Unresolved Questions about Standardization

All the previously described scientific fields solve problems that relates at least partly to standardization, on the side of the nomenclature $\mathcal{N}$ as well as on the side of the standardization function f itself. Apart from the unsupervised latent variable models, a discriminative common aspect of these automated approaches is to involve some external knowledge. Indeed, if standardization has been widely tackled by the literature, the only ready-to-use solutions are the multiclass classification and the entity linking. However, the former requires a labeled data-set of documents, while the latter requires knowledge base dedicated to the entities that we want to standardize. We will however never meet these ideal situations during this thesis, so that we have to study some questions that remain unresolved by the literature. As stated in Section 1.6, we will tackle standardization by decreasing order of the available external knowledge, which successively raise the following questions:

- The most classified documents that we need to standardize are the structured texts, which take values in a hierarchized nomenclature. Each website having its own nomenclature, we will see that standardizing is equivalent to schema mapping; however, the only strategy that applies to our standardization problem is the mapping reuse, poorly treated by the literature. The first question that arises is therefore the feasibility to leverage a large data-set of previous schema mapping, in order to reproduce automatically the semantic equivalences. Furthermore, to what extent such an automated schema mapping permits a reliable standardization of structured data?

- Some structured knowledge bases appears to be directly usable as nomenclatures, for example the national job categories. In the case of an abstract nomenclature, the problem of standardization is not solvable through entity linking. The example of jobs' categorization is even regularly treated as a multi-class classification for which the semi-supervised learning has shown promising results. In the case where we have only a limited data-set of standardized documents, is it feasible to leverage efficiently the categories' descriptions? Is it moreover possible to enrich the system with a corpus of unlabeled documents, as does the semi-supervised learning?

- The entity linking proposes to standardize a mention into a fixed knowledge base. However, the existing knowledge bases generally used are too generic for a domain-specific standardization. Is it feasible to automatically generate a domain-specific knowledge base for a standardization problem? What would be the general process for this, and how to evaluate the quality of the generated nomenclature just based on the final standardization? This evaluation is particularly critical when the generation relies on data matching, whose variants would produce nomenclatures of varying qualities.

- The latent variable models show the great advantage of not requiring any external structured knowledge, but just a corpus of documents. Such mod-

els are however not usable as standardization because the topics or clusters have no explicit meaning. They even change every time we learn the model. However, incorporating some meta-data for some problems appears to be a current prospect. Is it possible that a latent variable model leveraging some meta-data provides more meaningful clusters? Which latent variable model would therefore be usable in the case of sequences of documents, like the degrees of candidates?

These questions will be studied successively throughout the chapters of this thesis, starting with locally structured data.

Chapter 3

Mapping Local Nomenclatures of Structured Attributes

In order to adress automatic standardization, it appears necessary to leverage external structured knowledge. In the case of data extracted from websites, such as the documents involved in the SmartSearch project, many textual fields are already structured: they take values in the nomenclature of the source website. A first natural approach to standardization is therefore to make use of these local nomenclatures, which will be covered in this chapter. In other words, we will study the standardization of *structured* data (see Section 1.3.3), which are pre-standardized differently on each source. It is for instance the case for the contract type or the industry sector of a job advert, values that are selected by the recruiter in the website's nomenclature. As pointed out in 1.3, each website has a different nomenclature, which raises the common problem of interoperability between various sources. There is therefore a need for an a posteriori standardization, in order to translate the attributes in a common nomenclature. In this chapter we will see how standardizing this type of data is partially resolved by schema mapping, for which we proposed a case-based automation in [Malherbe et al., 2015b].

This study is also a more in-depth introduction to the standardization problem when dealing with different sources and vocabularies, and furthermore raises the difficulty of automatically reproducing semantic reasonings.

3.1 Standardization through Mapping of Nomenclatures

The job adverts as well as the candidate profiles involved in SmartSearch project are extracted from various recruitment websites, and a first natural strategy for standardization is to study the data that is structured on these websites. This is the case for the **Contract** type, the industrial **Sector**, the job **Function**, the required **Experience** and **Study** level for a job advert or an occupied position, manually specified by the recruiter or the candidate in the website's nomenclature. Such attributes are represented by textual documents, but take a finite

number of values for a given website (see the definition of structured data in Section 1.3.3). In light of this, if we translate these few values into a common nomenclature, then all the corresponding data coming from this website is standardized.

In this case, and only in this case, standardization can be addressed manually. Indeed, the millions of documents we manipulate come from only a few hundreds of recruitment websites (see Table 1.1 of Section 1.2.2), meaning that only few hundreds of local nomenclatures need to be standardized. The task is then to find for each value of the local nomenclature an equivalent in the target common nomenclature, which corresponds to mapping the nomenclatures as shown visually in the Figure 3.1. The choice of the common nomenclature, i.e the target of standardization, depends on the industrial context; in the SmartSearch project, we aim to have a generic nomenclature for each attribute, but we could imagine having personalized nomenclatures for each client using the software. In light of this, this chapter does not focus on a specific target nomenclature $\mathcal{N}$, but just proposes to reproduce the manually pre-defined mappings, as an exploratory study. Since a good part of the nomenclatures are hierarchized, we are in fact dealing with schemas (see Section 2.2.2); the problem we address is hence schema mapping, which is a recurrent approach for dealing with interoperability.

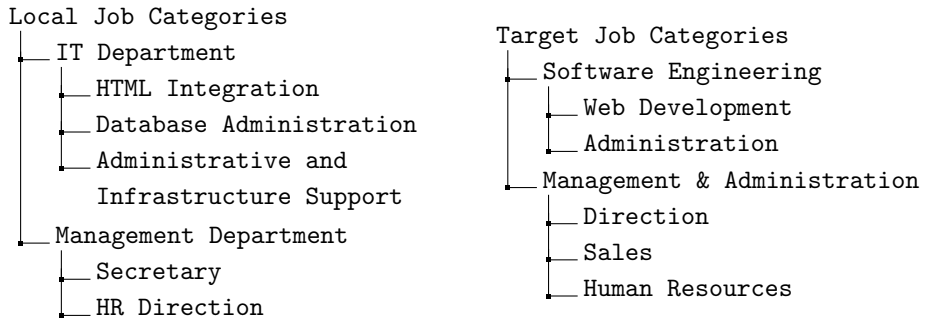


Figure 3.1 – Example of the schema matching problem : given the text schemas $A^{example}$ in the left-hand side and $Z^{example}$ in the right-hand side, what is, for each item in $A^{example}$ the best semantic equivalent in $Z^{example}$?

In practice, Multiposting faces this kind of standardization in another software, which serves at publishing automatically a job advert from the client to a selection of recruitment websites. In this example of industrial inter-operatibility, the job advert fields are structured in the client's internal nomenclature, and need to be standardized into the website's nomenclature before being published. Dozens of nomenclatures need to be mapped one to another each week, due to the constant increase of Multiposting's clients and supported websites. As an example, the Figure 3.1 could represent respectively Microsoft's and Monster.com's nomenclatures for job categories.

In term of formalized standardization, which we defined as a function $f(\boldsymbol{d}) \in \mathcal{N}$ in Section 2.1.4, the documents $\boldsymbol{d}$ take a finite number of values for locally structured data. When looking at one source of data, these values make up the lo-

cal nomenclature $\mathcal{N}' = \{\boldsymbol{d_1}, \boldsymbol{d_2}, ..., \boldsymbol{d}_{|\mathcal{N}'|}\}$,, and standardizing signifies giving the values of $f(\boldsymbol{d_1}), f(\boldsymbol{d_2}), ..., f(\boldsymbol{d}_{|\mathcal{N}'|})$ in the target nomenclature $\mathcal{N}$. The function f is therefore a map between $\mathcal{N}'$ and $\mathcal{N}$, meaning each concept of the left-hand side of Figure 3.1 is linked to its equivalent in the right-hand side. Handling more sources means doing new mappings from a nomenclature into another, which defines f incrementally.

In this chapter, the nomenclatures are supposed to be hierarchized, and are not just a flat set of values $\boldsymbol{d}$. For this reason, the standardization function f will be applied to what we call items, which will be defined in the next section as well as schemas.

3.2 Formal Definitions

3.2.1 Representation of Schemas and Items

We firstly introduce here the notion of a schema item, which corresponds to a concept in the nomenclature, or to a node if one considers the schema a tree. The item is for instance a contract type, a job sector or an experience level. To define an item, we consider the *name of the concept*, and the name of the *parent concept*, which are both textual documents. This parent name gives the semantic context of the item, and captures the hierarchy of the schema. For clarity, we will only focus on such 2-level items, but higher levels of the hierarchy could be taken into account, by considering for instance the name of the *grandparent node*. An *item* is therefore defined as a pair of documents $a = (\boldsymbol{a_1}, \boldsymbol{a_2})$, made up of the bag-of-words $\boldsymbol{a_1}$ of the concept, and the bag-of-words $\boldsymbol{a_2}$ of its parent. These bags-of-words are obtained following the process described in Section 2.1.1 using a stemmer for processing words, and $\boldsymbol{a_2}$ equals to the empty multiset $\{\!\{\}\!\}$ when the considered concept has no parent. If one considers the schema as a tree, $\boldsymbol{a_1}$ corresponds to a *node* of the schema, while $\boldsymbol{a_2}$ corresponds to its *parent*, as described in the example below. Grouping items in a set is another way of representing 2-level schemas.

This definition of items enables us to represent a schema as a set of items. For instance, the left-hand side schema in the Figure 3.1 is represented as the following items set:

$$
\begin{aligned}
A^{example} = \Big\{ & \Big(\{\!\{html,\ integration\}\!\},\ \{\!\{it,\ department\}\!\}\Big), \\
& \Big(\{\!\{database,\ administration\}\!\},\ \{\!\{it\ department\}\!\}\Big), \\
& \Big(\{\!\{administrative, infrastructure,\ support\}\!\},\ \{\!\{it,\ department\}\!\}\Big), \\
& \Big(\{\!\{hr,\ direction\}\!\},\ \{\!\{management,\ department\}\!\}\Big), \\
& \Big(\{\!\{secretary\}\!\},\ \{\!\{management,\ department\}\!\}\Big)\Big\}
\end{aligned}
$$

And similarly for the right-hand side schema of Figure 3.1:

$$Z^{example} = \Big\{ \Big(\{\!\{web,\ development\}\!\},\ \{\!\{software,\ engineering\}\!\} \Big), \\ \Big(\{\!\{administration\}\!\},\ \{\!\{software,\ engineering\}\!\} \Big), \\ \Big(\{\!\{direction\}\!\},\ \{\!\{management,\ administration\}\!\} \Big), \\ \Big(\{\!\{sales\}\!\},\ \{\!\{management,\ administration\}\!\} \Big) \}\!\}, \\ \Big(\{\!\{human,\ resources\}\!\},\ \{\!\{management,\ administration\}\!\} \Big) \Big\}$$

The term *schema* will equivalently denote the formal schema (on the top) and the corresponding items set. Hence, $Z^{example}$ will be referred to as a schema. The schemas will be written in capital letters, and we will write $a \in A$ to say that the item a is in the schema A.

3.2.2 Formal Schema Mapping

Given two schemas A, Z the general problem of schema mapping is to associate one by one each item $a \in A$ to its equivalent item z in the schema Z. This problem can be solved by considering successively the items $a \in A$; hence our schema to schema mapping boils down to solving several sub-problems, each of them being denoted by a triplet $pb = (a, A, Z)$. Sub-problem pb consists in finding the item z in schema Z that is the closest to the meaning of a in schema A. The solution z is therefore the value of the standardization function $f(a)$ in the schema Z. A will be referred to as the *initial schema* and Z as the *destination schema*. We then assume that the problem systematically has a solution: it is not true in theory, but it is an industrial constraint that every item in A finds an equivalent in Z. In practice, the schemas to be mapped describe similar knowledge and the assumption is verified.

The problem and its solution (a, A, Z, z) can be compared to the definition of mapping in [Bouquet et al., 2004]. Similarly to them, we will define in equation 3.3 of Section 3.3.1 a *score* that estimates the quality of the mapping, which corresponds to what they refer to as the degree of trust to the mapping. On the other hand, it is worth noticing that contrary to them, we explicit the schemas as part of the problem, and we consider our problem as *asymmetric*.

For instance, let us consider the problem of finding a semantic equivalent of the first item in the left in Figure 3.1 among the items of the schema in the right. If we write the first item as $a^{example} = (\{\!\{html,\ integration\}\!\},\ \{\!\{it,\ department\}\!\})$, this example problem is formalized as $pb^{example} = (a^{example}, A^{example}, Z^{example})$. The solution of this target problem would then be

$$(\{\!\{web,\ development\}\!\},\ \{\!\{software,\ engineering\}\!\})$$

which will be denoted $z^{example}$ and verifies $z^{example} \in Z^{example}$.

3.2.3 Data-Set of Past Mappings

As it is pointed out in the related work of Section 2.2.2, schema mapping is a task that is generally performed manually in practice. At Multiposting, the schemas involved in the automatic job postings have been mapped fully manually for years, as described previously in Section 3.1. Following the above formalization, this process requires the employee to select the solution z to the problem pb, which is particularly time consuming. A purely automatized standardization would be ideal but a computer assistance would already be a significant gain of time.

The set of all previously solved mappings are formally represented by entries (pb, z) where z was given as the solution of the problem pb. These solved mappings form a huge database, manually maintained over the years. This database contains numerous semantic equivalences between nomenclatures of different websites and companies' systems, which make up particularly relevant information in order to compute the standardization fonction f. The section 2.2.2 reveals that only little work related to schema mapping leverages mapping reuse. To tackle the problem, we propose to leverage the database through a Case Based Reasoning methodology [Watson, 1999], often abbreviated as CBR, which will be detailed in the next section.

3.3 Automatic Nomenclature Mapping: a First Attempt

3.3.1 Case Based Reasoning Steps

The CBR methodology bases its computation on a data-set $\mathcal{M}$ of solved mapping problems, that are called *source* cases m, which each represents a pair $m = (pb', sol')$, where the source problem $pb' = (a', A', Z')$ describes the mapping of a single element and $sol' \in Z'$ the retained solution. The CBR methodology consists in the following four standard steps [Aamodt and Plaza, 1994]:

- Retrieve: Given a *current* problem pb to be solved, a sub-set of *source* cases relevant to solving it are retrieved. We write a case of solved mapping, written $m = (pb', sol')$ to avoid confusion with the current problem pb.
- Reuse: The solutions sol' of the retrieved cases are used to propose a solution sol to pb, which possibly requires to adapt the solutions sol' to fit the constraints of the current problem.
- Revise: The proposed solution sol is tested in the real world and revised if necessary, which generally involves manual supervision.
- Retain: The final solution is kept in memory in order to add a new solved mapping m in the base $\mathcal{M}$.

We instantiated the retrieve, reuse and revise steps of schema mapping, which are illustrated from Bob's point of view in Figure 3.2 and are described in details

in the next sub-sections. This algorithm could also be described as a closest neighbor (see Section 2.3.1) that would classify positive mappings based on a custom distance function $d(m, m')$ defined between two mappings m, m'. For the sake of clarity and for justifying our similarity functions (given by Equations (3.2) and (3.3)), we prefer to dissect our system through a CBR methodology.

The Retrieve step

First of all, we define a function for comparing two items one to another. Given two arbitrary items a, a', the similarity between them is defined as:

$$g(a, a') = \frac{cos(\boldsymbol{a_1}, \boldsymbol{a'_1}) + cos(\boldsymbol{a_2}, \boldsymbol{a'_2})}{2} \quad \in [0, 1] \tag{3.1}$$

where the cosine measure *cos* between two documents was defined by Equation (2.4) of Section 2.1. All values computed by this function belong to the interval $[0, 1]$, with 1 corresponding to a perfect match. The left part $cos(\boldsymbol{a_1}, \boldsymbol{a'_1})$ computes the similarity between the leaves of each item, whereas the right part compares the parents.

For instance, with items introduced in Section 3.2.2, we have:

$$\begin{aligned} g(a^{example}, z^{example}) &= \frac{1}{2}\, cos(\{html,\ integration\},\ \{web,\ development\}) \\ &+\ \frac{1}{2}\, cos(\{it,\ department\},\ \{software,\ engineering\}) \\ &= 0 \end{aligned}$$

This example also shows us the need to go beyond a *direct match* between items, which leads here to a zero similarity, whereas the meanings of $a^{example}$ and $z^{example}$ are very close.

In order to determine what case is relevant for solving a new problem that occurs, we introduce an inter-problem similarity function $h(pb, pb')$. As a first attempt, it is simply defined as:

$$h(pb, pb') = g(a, a') \tag{3.2}$$

which forgets all information about the initial schema A. When we have $h(pb, pb') = 1$, it implies that: $a = a'$. One notes that this similarity only takes into account the item a, forgetting schemas A, Z when retrieving useful cases. We will propose similarity functions in the section 3.4 that address this limit.

Thanks to this function, for every new problem pb that occurs, we can determine which existing case $m' = (pb', sol')$ stored in the base is the most useful to help us solve pb. Once retrieved, we need to exploit the information provided by its solution, as explained in the next subsection.

The Reuse step

In this step, we want to exploit the most similar case found during the *Retrieve* step. However, if we consider a source problem $pb' = (a', A', Z')$, we do not

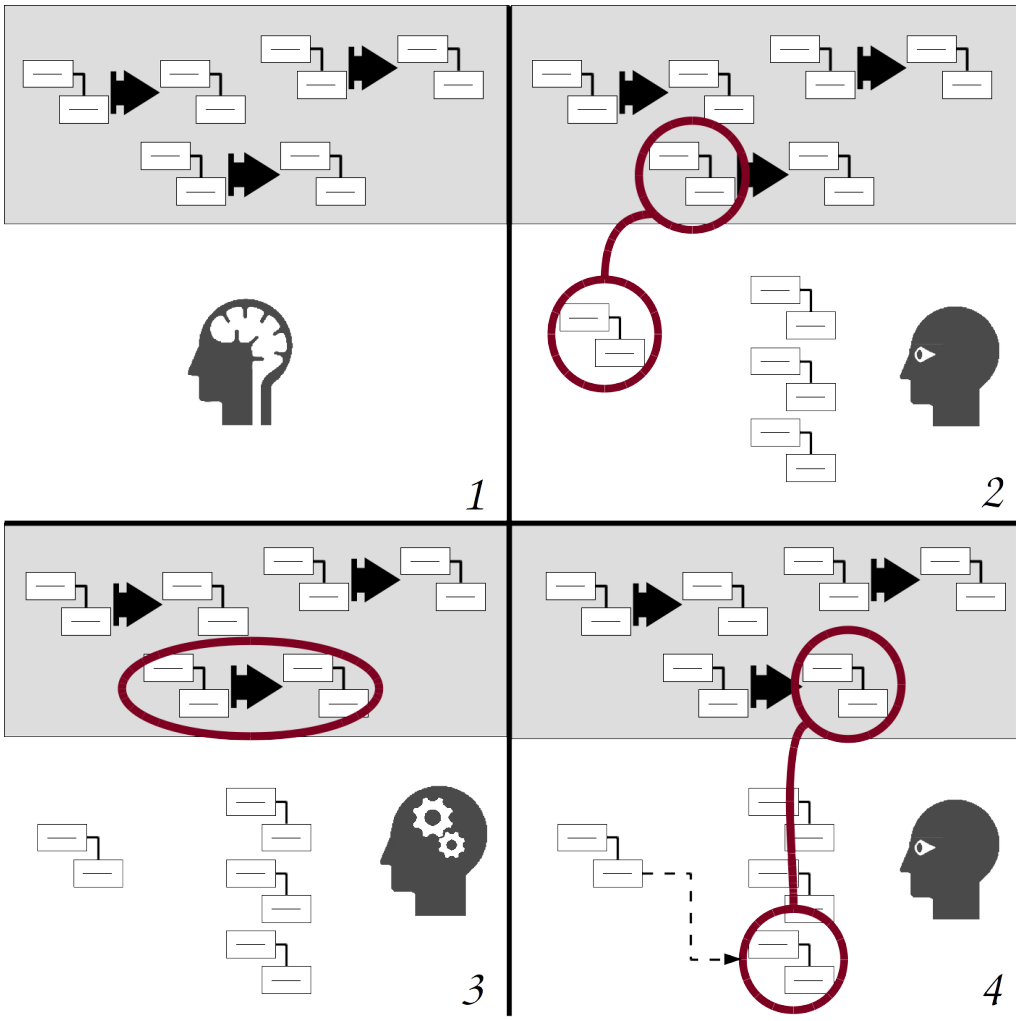


Figure 3.2 – The CBR approach to schema mapping as perceived from Bob's point of view. He firstly memorizes in the first frame the base of previously made mappings, illustrated here just with pairs of items $a' \rightarrow sol'$ for clarity. In practice, this memorization is done incrementally through the successive retain steps. Then, in the second frame, given the current problem illustrated by the item a and the target schema Z, Bob retrieves the most relevant case, based on the similarity between a and a'. In the third frame, Bob considers the solution sol' memorized for this retrieved case. In the fourth frame, Bob then takes among the possible current solutions the most similar to this solution sol'. In this example, only one case is used for simplicity, but the system described in this chapter considers k retrieved cases. In practice, the choice of the solution is manually revised a posteriori by an expert.

necessarily have $Z = Z'$. In particular, $sol' \notin Z$, which means that we can not take the corresponding solution sol' as our current solution; we thus need an adaptation to find a solution in $sol \in Z$.

In light of this, we propose to rank the possible solutions z of the current destination schema Z. In order to use more information to rank possible mappings, not just one but the k most similar cases are retrieved from the base ($k = 100$ gave satisfactory results in our experiments). Let us denote M_{pb} the subset of $\mathcal{M}$ that contains the k most similar cases m' to the problem pb, with respect to the similarity function g.

Given the current problem $pb = (a, A, Z)$, a ranking of the possible solutions $z \in Z$ is computed using the function $match(pb, z) \in [0, 1]$ defined as:

$$match(pb, z) = \max_{m'=(pb', sol') \in M_{pb}} h\left(pb, pb'\right) \times g\left(z, sol'\right) \tag{3.3}$$

The idea is to find a source problem pb' similar to pb such that the corresponding solution sol' is similar to item z - the first similarity being evaluated by the function h (problems similarity) and the second one by the function g (items similarity).

The Revise step: Ranking Suggestions to the User

After the previous step, the top ranked item $z \in Z$ that maximizes $match(pb, z)$ could ideally be taken as the final solution. However, this ready-made approach would likely provide poor results because of the fact that distinguishing a good from a bad mapping is a matter of semantics and is thus difficult to fully automatize. Furthermore, any CBR system aims to improve its problem solving capacity over time, and the automatic approach would likely fail to enhance the system's performance.

Consequently, a manual revision is necessary in our system, but it is greatly eased by the match scores computed in the reuse step. The *Revise* step consists thus in asking the right mapping to the user, which is made as fast and easy as possible thanks to the previously computed ranking. Given the problem $pb = (a, A, Z)$, items z of Z are ranked with respect to $match(pb, z)$, and the user chooses manually which element of Z corresponds to a. The ranking of all elements of Z helps a lot, especially when the cardinality of Z is big.

3.3.2 Preliminary Evaluation

Let us first describe the experimental data-set of mappings $\mathcal{M}$. As explained in Section 3.1, the schemas mapped at Multiposting are the nomenclatures of the e-recruitment websites. They are in different languages, mainly French, English, German, Spanish and Polish. They represent different kinds of information, and a good part of them have been manually grouped into 6 types corresponding to the job structure of Section 1.3, for example the "contract type" and "job category". Moreover, many nomenclatures are specialized, and correspond for instance to some university degrees or to a recruitment website for managers. This diversity of schemas should be handled by our algorithm, that retrieves

only the useful cases. Characteristics of the data-set are listed in Table 3.1. Note that we also have an "unknown" schema type in our data-set for uncategorized schemas. The same table also gives statistics for the full data-set of mappings $\mathcal{M}$, resulting from the aggregation of all categories.

Schema Type	All	Contract	Study	XP	Location	Sector	Category
# schemas	2,257	184	94	69	21	205	296
# terms	99,875	713	1736	288	36,960	5,120	12,541
# items/schema	87	7	23	7	3,152	55	227
Schema mappings	3,356	182	96	71	18	340	691
Item mappings	215,701	1,330	839	629	2,017	19,926	88,581

Table 3.1 – Characteristic from our data-set, by schema category: number of terms, average *number of items per schema, number of item pairs mapped, of schema pairs mapped and number of schemas.*

In order to evaluate the performance of our CBR approach, we used a cross validation on the data-set. The Leave-one-Out Cross Validation is commonly used for testing a CBR algorithm [Gu and Aamodt, 2006], and more generally for machine learning algorithms. To cross validate our system, at each fold, we take out a schema mapping from the case base, and we run the algorithm on the excluded mapping to see the ranks of the manually selected items. From this cross validation, we deduced the relevancy of our system's suggestions. In order to do so, we compute the precision at k which is a metric from information retrieval systems [Büttcher et al., 2010]:

$$P@k = p(\text{solution is in the k top ranked items}) \in [0, 1]$$

where this probability is given by the number of test mappings for which the algorithm had ranked the valid solution in the top-k suggestions, divided by the number of test mappings. One notes that the precision at k increases with k; in our experiments, we computed $P@k$ for $k = 1..10$.

The computed precisions $P@k$ are displayed in Figure 3.3, with k as abscissa. From this figure, we can conclude that the case-based reasoning applies to our problem; for instance, in 70% of the runs, the validated mapping has been ranked in the 5 first suggestions. However, the performance seems still quite low, since the algorithm firstly suggested a correct mapping in only 36% of the cases. This confirms that we can not imagine a totally automatic CBR taking the mapping with the top-ranked item as a solution.

Figure 3.4 shows the precision of the first suggestion $P@1$ for the different types of schemas mapped. The geographical location and contract are the types with highest precision. These problems are indeed the most simple, since these schemas are nomenclatures of entities and the mapping mainly maps synonyms. On the contrary, job sectors and job categories are more difficult to map, since they are abstract nomenclatures; the problem is then to find semantic equivalents, that can be synonyms, but also hyponyms or hypernyms, for instance. As it could be expected, the sectors, which are less precise than categories, are better suggested by our CBR.

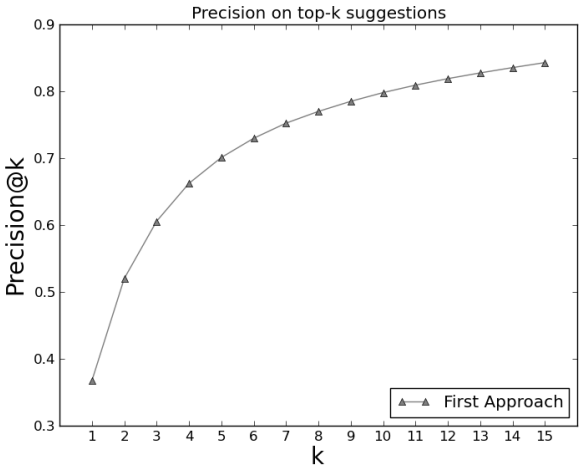


Figure 3.3 – Precision on top-k suggestions for this first approach.

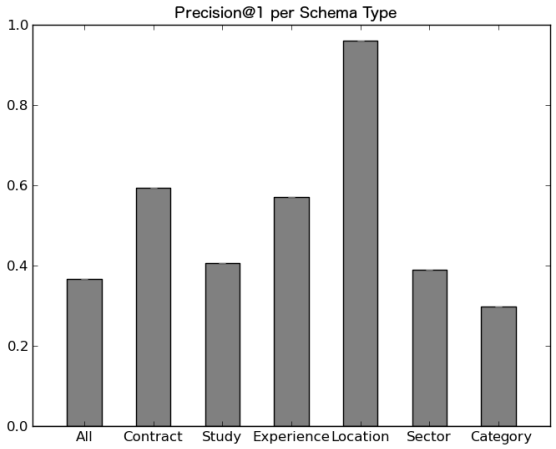


Figure 3.4 – Precision on the first suggestion, by schema type.

3.4 Improvement by System Tuning

In this section, we explore some improvement and their effect on the system's performances. In the previous section, we made arbitrary choices concerning the definition of the inter-problem similarity $g(pb, pb')$, and the function $match(pb, z)$ used in the ranking. In the following, we propose more sophisticated definitions in order to make the CBR system more precise in its suggestions.

3.4.1 Alternative Inter-problem Similarity

First of all, let us introduce an operation to compare not only items but whole schemas one to another. As stated in Section 3.3, we did not exploit the schemas A or Z to define the inter-problem similarity. Let $\boldsymbol{cat}(A)$ be the bag-of-words obtained by concatenating all the documents of the schema A, with deduplicated terms. $\boldsymbol{cat}(A)$ conveys therefore crucial information, which is the vocabulary contained in A. For example, with the schemas introduced in Figure 3.1,

$\boldsymbol{cat}(A^{example})$ is the following bag of words:

$$\{\!\{html,\ integrator,\ it,\ department,\ administrative,\ infrastructure,\ support,\ database,\ administration,\ hr,\ direction,\ management,\ secretary\}\!\}$$

and $cat(Z^{example})$ is the bag of words:

$$\{\!\{management,\ administration,\ software,\ engineering,\ web,\ development,\ direction,\ administration,\ sales,\ human,\ resources\}\!\}$$

Thanks to this new definition, we can compute by $cos(\boldsymbol{cat}(A), \boldsymbol{cat}(A'))$ how similar the two schemas A and A' are, based on their terms in common. We then refine the inter-problem similarity to take into account the destination schema Z, which is the set of possible solution items:

$$h(pb, pb') = g(a, a') \times cos(\boldsymbol{cat}(Z), \boldsymbol{cat}(Z')) \tag{3.4}$$

This similarity will be referred to as *"right vocabulary similarity"*, because $cat(Z)$ represents the vocabulary of Z, and the destination schema Z is represented in the right hand side. For instance, with the example problem $pb^{example}$ defined in Section 3.2 and $pb'^{example} = (z^{example}, Z^{example}, A^{example})$ we have:

$$\begin{aligned} h(pb^{example}, pb'^{example}) &= g(a^{example}, z^{example}) \times f(\boldsymbol{cat}(A^{example}), \boldsymbol{cat}(Z^{example})) \\ &= 0 \times 0.15 \end{aligned}$$

where the destination schemas $A^{example}$, $Z^{example}$ have terms in common, namely *Management* and *Administration*.

For comparison purposes, the similarity function h defined in the first solution in Section 3.3 will now be referred to as *"no vocabulary similarity"*. Similarly, we will try other variants of the similarity function, starting with the *"left vocabulary"* similarity which considers the initial schema A:

$$h(pb, pb') = g(a, a') \times f(cat(A), cat(A')) \tag{3.5}$$

and the *"both vocabularies similarity* that considers both schemas A, B:

$$h(pb, pb') = g(a, a') \times f(cat(A), cat(A')) \times f(cat(Z), cat(Z')) \tag{3.6}$$

Those variants are proposed so that the system will take into account the context, when performing the schema mapping. The first solution proposed in this article indeed uses a very basic inter-problem similarity based only on the inter-item similarity g between the items a, a' that we want to map. This does not take into account the context of the problems that we compare in the retrieve step. Indeed, the destination schemas Z, Z' may be totally different even if a and a' are very similar. In practice, this phenomenon occurs when two problems with almost identical items a, a' are applied to schemas in different languages, e.g. English and in French. Similarly, mapping an item to a list of job sectors, or to a list of university courses are also two very different problems, which is expressed by the global difference between Z and Z'.

3.4.2 Alternative Score Function

When considering a source mapping $m = (pb', sol')$ of the case base $\mathcal{M}$, the retained solution sol' has been initially validated by a human expert. Since human operators are prone to make mistakes, we want to prevent our CBR from any overfitting on the errors contained in $\mathcal{M}$. Indeed, the score function proposed previously in Equation 3.3 is based on the single case that maximizes inter-problem similarity, so that a single bad mapping can influence further responses negatively. In other words, a single flawed case can change score values dramatically and the system may fail in helping the user in the very long run. On the other hand, we can hope the best solution z for problem pb to have several similar cases in the base; we therefore propose to leverage not one, but 5 similar cases in the match score used for ranking the solutions. We define an extended match score as:

$$\begin{aligned} match'(pb, z) \;=\; & \max_{m'=(pb',sol')\in M_{pb}} h\left(pb, pb'\right) \times g\left(z, sol'\right) \\ & + \; 2^{nd} \max_{m'=(pb',sol')\in M_{pb}} h\left(pb, pb'\right) \times g\left(z, sol'\right) \\ & + \; \ldots \\ & + \; 5^{th} \max_{m'=(pb',sol')\in M_{pb}} h\left(pb, pb'\right) \times g\left(z, sol'\right) \end{aligned} \tag{3.7}$$

This new function hence computes a ranking based not only on the most similar case, but on the five most similar cases. Using several cases at a time, the system becomes a lot less human error-prone. The new match score returns meaningful results, even if a bad mapping is unintentionally saved in the base of mappings $\mathcal{M}$.

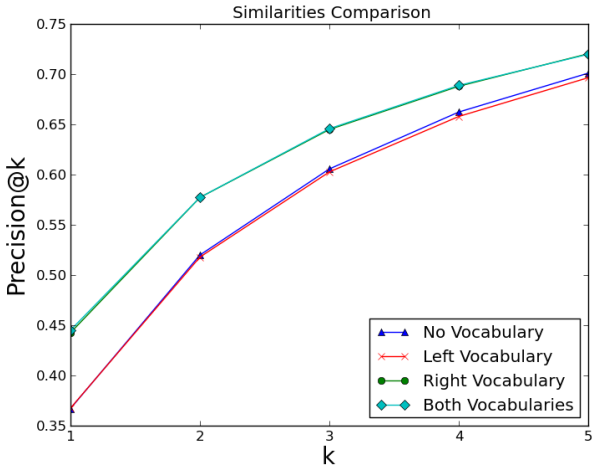


Figure 3.5 – Precision of the system using three different inter-problem similarity functions proposed in Section 3.4.1. The curves for the "no vocabulary" and "left vocabulary" similarities are almost indistinguishable, as well as the curves for "right vocabulary" and "both vocabularies" similarities.

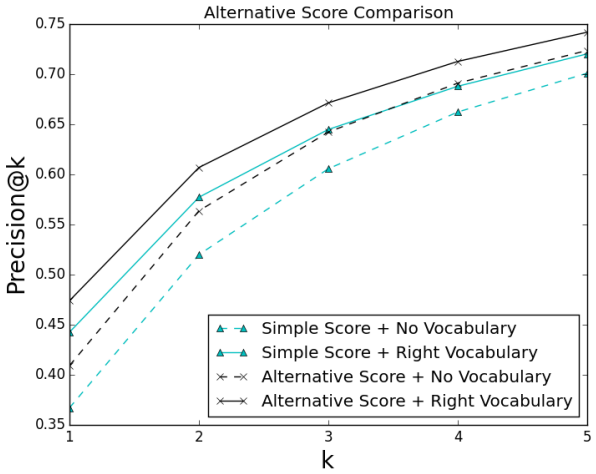


Figure 3.6 – Precision of the system using two different score functions defined in Sections 3.3.1 and 3.4.2, with and without right vocabulary inter-problem similarity.

3.4.3 Overall Evaluation

Figure 3.5 shows results when we try variants of similarity measures defined in Equations (3.2), (3.4), (3.5) and (3.6), with a simple revision step using the match score of Equation (3.3). At first sight, we see an improvement when the right vocabulary is taken into account, with almost a 10% increase for the precision at the first suggestion $P@1$. Second, we note that taking into account the left vocabulary is useless: the curve obtained when the left vocabulary is included in the similarity has exactly the same shape. In light of this, a problem pb of schema mapping should be only represented by (a, Z), since it depends on the possible solutions Z but not on the neighbor items of a contained in the initial schema A.

Figure 3.6 shows the results for the extended revision step, with the simple match score defined in Equation (3.3) and with the extended score defined in Equation (3.7). We only compared algorithms with simple *"no vocabulary similarity"* - Equation (3.2) - and *"right vocabulary similarity"* - Equation (3.4). In both cases, the extended revision step increases accuracy by about 5% in general, it is therefore relevant to consider several similar cases in the scoring. This could be because of the poor quality of some cases in $\mathcal{M}$, which would lead to biased suggestions.

We decided to implement the system tuned with the alternative match score of Equation (3.7) and the *"right vocabulary similarity"*. In the interface shown in Figure 3.7, items on the right side represent the schema Z of possible mappings, and are ranked with respect to $match'(pb)$. In order to ease visualization, the items are associated to a color (gradients of blue) proportional to $\frac{match'(pb,z)}{\max_z match'(pb,z)} \in [0, 1]$. The top-ranked item is associated to a correctness, which estimates how reliable the first ranked solution is. Displayed on the top-right cor-

Figure 3.7 – Interface of the computer-assisted tool for schema mapping. The current item to be mapped is highlighted in green on the left-hand side, and the items on the right-hand side are ranked with respect to their score.

ner in the interface, it is estimated by a supervised learning on the data-set $\mathcal{M}$ following a method detailed later in Section 4.5.

Evaluation	Cross validation		Users' usage on new mappings
System	Initial system	Tuned system	Tuned System
Average rank	2.56	2.14	1.72

Table 3.2 – Average rank of the solution in the computer-assisted system, computed by cross-validation on the initial data-set or on user's real-world usage.

After two years of use, the users are satisfied with this tool, and the time spent on schema mapping has significantly decreased: 62% of items have been mapped by a clic on one of the top suggestion, 81% on the top 3, whereas in the unranked case the user had to look at dozens of items (87 on average, see table 3.1). Based on the real-world usage since the release of the interface, we computed the average rank of the user's choice, sometimes denoted by mean rank [Büttcher et al., 2010]. The users' choices might be biased by the predicted ranking, therefore in Table 3.2 we compare this value with the average rank computed by cross validation on the initial data-set, for the system described in Section 3.3 and the tuned system proposed in this section. The figures show how helpful the ranking is, and we evaluate the saving time per item mapping at 5 seconds. As there are about 30 schemas mapped every week with 87 items per schema on average, the time saved by the employees using our system is approximately 3 and a half hours every week.

Despite this positive feedback from the users, this standardization solution for locally structured data still requires manual action in practice. This is a serious limit for a scalable standardization, which should be automatic in order to process thousands of job adverts and candidate profiles every day.

3.5 The Necessity of a Generic Standardization Process

In the objective of a fully automatic system of standardization, the case-based system proposed in this chapter gives a satisfactory precision $P@1$ for some types of schemas, namely the required *Experience*, the *Contract* and the *Location* of a job advert, or a professional experience of a candidate. In these cases, standardization can be handled automatically or with very limited cost using the proposed system. This solution just uses the corresponding values that are originally structured in the source's nomenclature, and no additional information about the job advert or the candidate profile is needed.

This standardization approach could moreover be extended to some unstructured data as input, since the system does not need the considered item a to be originally taken from a schema A - the initial schema is indeed not used by our final solution. One could therefore consider a simple document $\boldsymbol{d}$ as input of our system and define the standardization function f as:

$$f(\boldsymbol{d}) = \underset{z \in Z}{argmax}\ match(pb, z)$$

where $pb = (\boldsymbol{d}, Z)$ represents the text $\boldsymbol{d}$ to be standardized - possibly with two levels, as items - and Z the target nomenclature. However the precision $P@1$ obtained for abstract nomenclatures is not high enough to automatically consider the top ranked item as a solution. Moreover, even when focusing on nomenclatures of entities, the studied source websites mainly use coarse-grained nomenclatures, so that the presented approach would not give good results for fine-grained nomenclatures of entities.

As a consequence, in order to infer fine attributes such as the category or the skills of a job advert, we can not leverage the locally structured fields that most sources provide. In other words, in the rest of this thesis, we will discard this rich knowledge, namely the structured values in the local nomenclatures, and prefer to infer attributes from unstructured raw texts, for example the *Title* and *Description* of a job advert. In the following chapters, the standardization function f will always take as input a raw document $\boldsymbol{d}$, regardless of its source website - contrary to the present chapter which addressed the standardization problem nomenclature by nomenclature. In light of this, we can still leverage structured external knowledge, but on the side of the nomenclature $\mathcal{N}$ instead of the side of the document $\boldsymbol{d}$. For instance, we will make use in the next chapter of the job category descriptions, made public for national nomenclatures.

Chapter 4

Categorizing Jobs using Category Descriptions

In this chapter, we study standardization into a fine-grained abstract nomenclature. As seen in the previous chapter, such standardization can not be handled using the local nomenclatures; moreover, the related work of Section 2.3.1 suggests that when $\mathcal{N}$ deals with abstract concepts, standardizing requires to find patterns in the document $\boldsymbol{d}$ through machine learning, instead of simply finding the mention of an entity as in the entity linking. In light of this, the literature mainly proposes to address this problem using supervised or semi-supervised multi-class classification, which both require a large data-set of standardized examples. We propose instead to cope this standardization with only a limited data-set, and efficiently leverage the external knowledge represented by the category descriptions associated to the nomenclature $\mathcal{N}$.

The object of this study will be the jobs' categorization, that has never been tackled by leveraging the external structured knowledge given by the existing rich nomenclatures. In light of this, we study in this chapter the use of the job category descriptions, that are made public for national nomenclatures. This rich knowledge appears to be crucial for standardization, as it shows a structure, is regularly updated and present relevant semantic information. However, several questions arise: How to efficiently find the proper category for a given job advers, since their textual content is not directly comparable? In particular, an special focus will be to use supervised learning but on a limited data-set of categorized job adverts. Beyond this aspect, how to follow the swift evolution of professions, whose most visible effect is the radical and sudden change in the utilized vocabulary? Last, how to estimate the probability for a categorization to be correct?

This chapter regroups the models we presented in [Malherbe et al., 2015c, Malherbe et al., 2015a, Malherbe et al., 2014]. They all have been positively implemented in the industrial environment represented by Multiposting.

4.1 A Nomenclature with Textual Descriptions

The professional category is particularly central in the e-recruitment platforms, since it leads the user in an effective exploration of the job adverts through a set of predefined categories. This categorization is an example of standardization into an *abstract* nomenclature: the job category is not explicitly named in a job advert, but is represented by several aspects of the job, for example the tasks involved or the required skills. This standardization appears to be impossible by using the original category of the job in its source nomenclature, as studied in chapter 3. One reason is that this attribute is specified by the recruiters in very heterogeneous nomenclatures: they are as many as websites, can be coarse-grained with only a dozen of categories, or really fine-grained with up to hundreds of categories. The objective of standardization enforces us to use a generic fine-grained nomenclature, and the corresponding job categorization would be purely based on the textual data of the job. The first natural strategy for categorizing a job is to use a multi-class classification algorithm on the job's textual data, that has been successfully applied to texts (see chapter 2). However, building a large and representative data-set of manually categorized job adverts would require extremely high human costs, especially if we want the nomenclature to be fine-grained - with typically hundreds of categories - and regularly updated.

Another strategy for standardization comes naturally when studying the possible nomenclatures. Indeed, there exist national nomenclatures that show great advantages: they aim to be generic, they are fine grained, and regularly updated by the national employment services. Another interesting aspect of these national nomenclatures are the job category descriptions, generally made public. As previously stated in Section 2.2.1, these textual descriptions are written by domain experts, for example the ROME description in Figure 4.1, and can be positively used for standardizing and indexing the job adverts. Using such detailed descriptions, the problem of standardization is represented to Bob as in Figure 4.2: the document representing a job needs to be put in one box, where each box is associated to a detailed category description. The strategy we adopt is then to match the job document to each category description, by computing a similarity score that estimates how well the job adverts corresponds to the category.

With the notations of Section 2.1.4, the nomenclature $\mathcal{N}$ is made up of objects n presenting a detailed description, that we propose to use by computing a similarity function $sim(\boldsymbol{d}, n)$ with the document $\boldsymbol{d}$ to standardize. The standardization function f assigns then a value to $\boldsymbol{d}$ according to:

$$f(\boldsymbol{d}) = \underset{n \in \mathcal{N}}{argmax}\ sim(\boldsymbol{d}, n)$$

where the similarity function $sim(\boldsymbol{d}, n)$ can be computed using machine learning, as we propose for our job categorizer. In the following, we firstly present the formalization of the job categorization problem, a preliminary similarity measure and the strategy for designing the final system.

Études et développement informatique (ROME : M1805)

Appellations métier

- Développeur / Développeuse Big Data
- Développeur / Développeuse d'application
- Développeur / Développeuse full-stack
- Développeur / Développeuse informatique
- Développeur / Développeuse - jeux vidéo
- Développeur / Développeuse multimédia
- Développeur / Développeuse web
- Développeur / Développeuse web mobile
- Développeur(euse) de sécurité des systèmes d'information
- Développeur(se) décisionnel - Business Intelligence
- Didacticien / Didacticienne informatique
- Homologateur / Homologatrice logiciel

Définition

- Conçoit, développe et met au point un projet d'application informatique, de la phase d'étude à son intégration, pour un client ou une entreprise selon des besoins fonctionnels et un cahier des charges.

Activités	Compétences
Intervenir dans un domaine informatique : • Embarqué • Informatique de gestion, Informatique décisionnelle - Business Intelligence • Industriel • Multimédia • Réseaux • Scientifique, Technique • Télécoms	• Langage Java • J2EE • J2ME • Langage C/C++ • Langage Pascal • Langage Ada • Langage C# • Langage Cobol • Langage Fortran • Langage HTML • Langage Lisp • Langage Perl • Langage PHP • Langage SQL • Langage Visual Basic • WLangage
Intervenir dans un domaine : • Finance, comptabilité • Ressources humaines • Logistique • Marketing • Production	

Figure 4.1 – Example of category description, with the first fields' contents: its Title*, the related job titles called* Labels*, its* Definition*, and the required* Skills *and* Tasks*. Other fields, out of 14 in total, are skipped for clarity, for example the working* Environment *and the required* Studies*.*

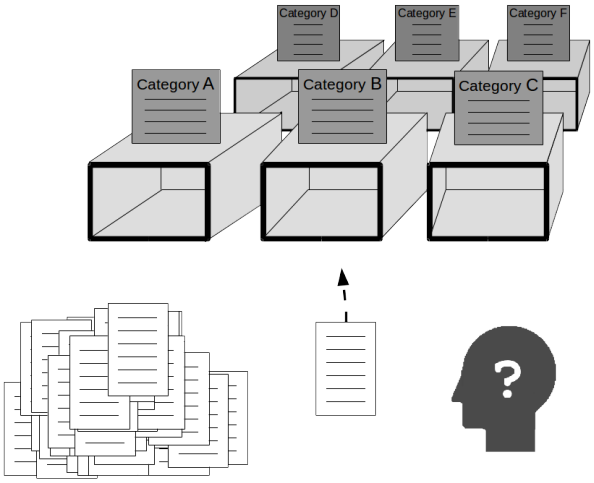


Figure 4.2 – The problem of job categorization as perceived by Bob. The descriptions over the boxes as well as the sheets of papers are separated into several fields, which are not visually shown here for clarity.

4.2 Preliminary Approach

4.2.1 Formal Representation of Jobs and Categories

In this chapter, we will make use of two different types of textual content: the job adverts and the job category descriptions. As explained in Section 1.3, a job advert is represented by different textual fields, among which we only consider the *Title*, the *Description*, the *Company Description* and the *Profile*, that are each free texts and present in almost any job advert, whatever the source. We formalize the content of a job advert j as a list of documents $\boldsymbol{j_i}$, each one representing a textual field:

$$j = (\boldsymbol{j_1}, \boldsymbol{j_2}, ..., \boldsymbol{j_{n_j}}) \tag{4.1}$$

where n_j is the total number of textual fields in a job advert j, (fixed to 4 in this study), and $\boldsymbol{j_i}$ is a multiset of terms, using the representation of documents of Section 2.1. In this study, the terms are stemmed using the implementation of [Porter, 1980], and some predefined stopwords are removed.

Similarly, a job category description (also simply called category) is defined by the textual description of a specific category of professions. This description being separated into several textual fields, we formalize the content of a job category c as a list of documents:

$$c = \{\boldsymbol{c_1}, \boldsymbol{c_2}, ..., \boldsymbol{c_{n_c}}\} \tag{4.2}$$

where n_c is the number of textual fields in the job category description, and $\boldsymbol{c_k}$ is a multiset of terms for the k-th field of the job category (with $k < n_c$). In this study, n_c is fixed to 14, whose full list of fields are readable in Figure 4.7. The set of the categories is written $\mathcal{C}$, with a fixed size of 531.

The above formalization can be linked to standardization, that we previously defined by a function f with values in a nomenclature $\mathcal{N}$. In this study, the set of categories $\mathcal{C}$ forms the nomenclature (previously written $\mathcal{N}$), and the function f takes as input a job advert j. This input is not exactly a text but a list of texts. In the following, we introduce through a simple model the strategy of ranking categories for a given job advert.

4.2.2 Direct Matching for Ranking Categories

The core of our approach is to rank the categories c given the job j to categorize, using a similarity function $sim(j, c) \in \mathbb{R}$. A natural approach is to use an existing similarity measure (see chapter 2.1), for instance the cosine measure *cos* (Equation (2.4)) of Section 2.1), and to hope that the textual contents of a category and a job are directly comparable. Since such measures take as input two classical documents, we need to build a unique document for the job advert and for the category, noted respectively $\overline{\boldsymbol{j}}$ and $\overline{\boldsymbol{c}}$, by concatenating the bag-of-words of each fields content:

$$\overline{\boldsymbol{j}} = \bigcup_{i=1}^{n_j} \boldsymbol{j_i} \qquad \overline{\boldsymbol{c}} = \bigcup_{k=1}^{n_c} \boldsymbol{c_k}$$

Since the category descriptions remain constant, contrary to the jobs, we consider for the vocabulary $\mathcal{V}$ the set of terms contained in all the job categories $\overline{\boldsymbol{c}}$. For each term t of $\mathcal{V}$, we compute the global term weight $idf(t)$ (Equation (2.1.2) of Section 2.1), based on the document frequency for the *category descriptions*, which form a corpus of 531 documents $\overline{\boldsymbol{c}}$. The motivation for computing the *idf* on the categories side is that a term is relevant if and only if it is infrequent in job categories, meaning it represents precisely a category - or inversely.

Using this $idf(t)$ and the term frequency count $tf(\overline{j}, t)$, we can compute the terms' weights for a document, like for $\overline{\boldsymbol{j}}$:

$$w_{\overline{\boldsymbol{j}},x} = \sqrt{tf\left(\overline{\boldsymbol{j}}, t\right)}\ (1 + log(idf(t))) \tag{4.3}$$

One notes that some terms t in the job adverts will have a weight of 0 (or, equivalently, will be absent from this formalization), since the vocabulary used $\mathcal{V}$ is restricted to the terms in the job category descriptions.

As a preliminary approach, given a job j, we propose to compute the similarity $sim(j, c)$ with category c as:

$$sim(j, c) = cos\left(\overline{\boldsymbol{j}}, \overline{\boldsymbol{c}}\right) \tag{4.4}$$

To categorize the job j, we simply select the category that maximizes the similarity, in other words the top-ranked one:

$$c^* = \underset{c \in \mathcal{C}}{argmax}\ sim(j, c) \tag{4.5}$$

This first approach is referred in the experiments as the Basic Cosine. As we will see later, it gets a relatively poor accuracy, but not ridiculous, with more than 60% job adverts of our data-set properly categorized: this preliminary similarity $sim(j, c)$ validates the approach of ranking the categories for standardizing a job advert's category.

4.2.3 Perspectives for Job Categorization

This preliminary approach permits to discuss more deeply about job categorization. Indeed, it shows some limits and is obviously improvable. The final system we propose is illustrated in Figure 4.3, and aims to address the following limits of the preliminary approach:

- The similarity *sim* looses the information about **the structure of the documents**; indeed, a word will be equally treated if it comes from the *Title* or the *Company Description* of the job, for instance. However, we have the chance that the templates for the job adverts and for the category descriptions are respectively static, so that we can leverage them. Since it is not trivial to efficiently match semi-structured documents, we develop in Section 4.3 a learning-to-rank approach based on a small annotated data-set.
- The matching relies completely on the category descriptions $c \in \mathcal{C}$ and the corresponding vocabulary $\mathcal{V}$. However, the job market is evolutive, and the official category descriptions might be outdated at a given time. Besides that, the descriptions might be incomplete and might lack some vocabulary. Last, these descriptions are not directly comparable to a job advert, since automatic categorization is not their original purpose. We propose therefore to **enrich the category descriptions** with new keywords in Section 4.4, using a massive data-set of unlabeled job adverts.
- The approach by ranking the categories does not provide **any probability estimate**: Taking directly the similarity $sim(j, c)$ is a poor estimate that the category c is correct for job j. However, it is very useful to estimate precisely the correctness of the system, especially for an industrial use. The section 4.5 proposes a generic approach for estimating the probability that a pointwise learning-to-rank based system is correct.

4.3 The Field-to-Field Similarity Model

This section starts by describing a model that leverages the structures of jobs and category descriptions. Then, we detail how the parameters of the model are learned, and the feature selection that results. The presented model is usable for matching any type of documents having a static template; for example, we successfully used it for ranking facebook profiles with respect to job adverts in [Malherbe et al., 2014].

4.3.1 Field-to-Field Similarity Model (FtFw)

In order to compute efficiently the similarity $sim(j, c)$ between the job j and category c, we propose to leverage their separation into different textual fields $\boldsymbol{j_i}$ and $\boldsymbol{c_k}$. With this objective, we assume that it is relevant to compare the i-th job field, represented by $\boldsymbol{j_i}$, to the k-th category field $\boldsymbol{c_k}$ and not to the

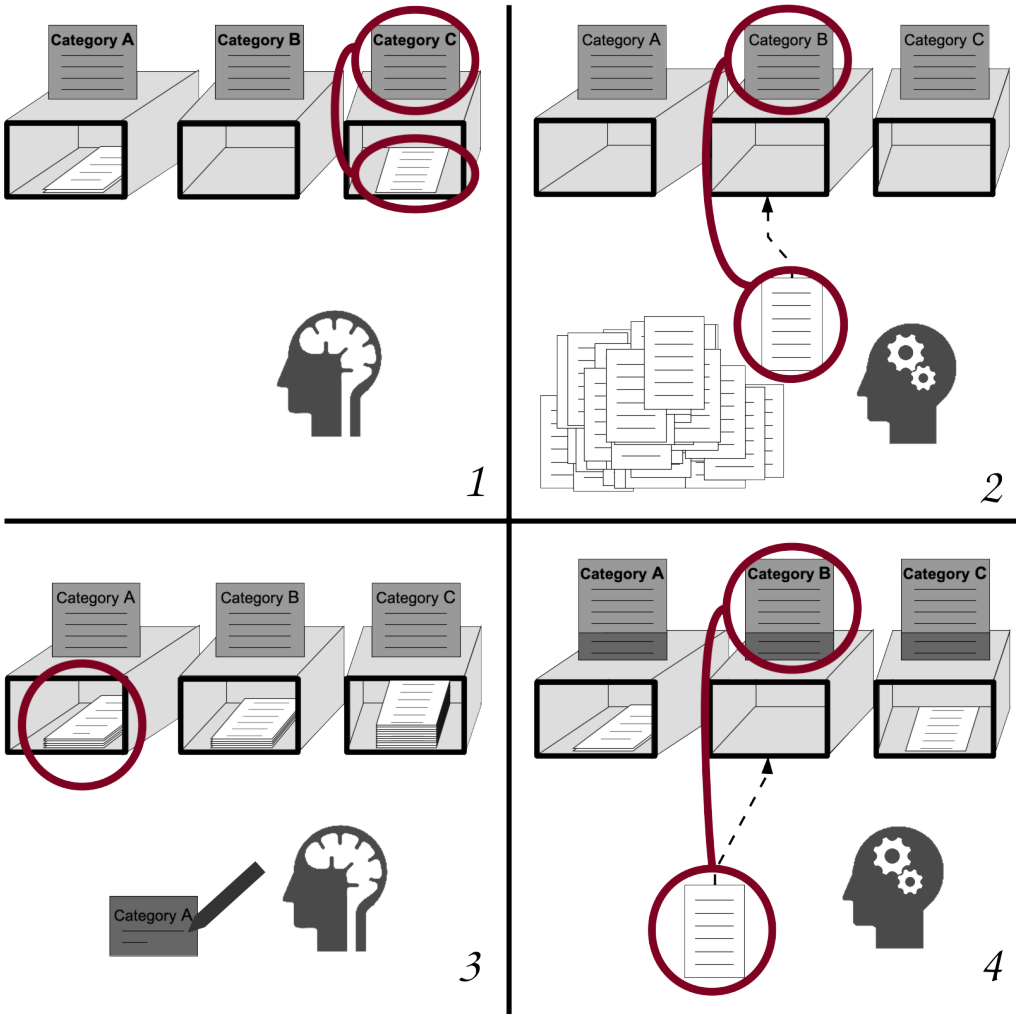


Figure 4.3 – The job categorization as Bob perceives it: in frame 1, he learns the Field-to-Field model (Section 4.3), for which he just needs to inspect some examples of pre-classified sheets of paper, but each box does not need to be filled with sheets. Using this model, Bob predicts the categories of a corpus of unlabeled job adverts in frame 2. By looking at the terms present in these artificially categorized jobs, Bob enriches the category descriptions in frame 3 (Section 4.4). The final categorization of a job, in frame 4, is a prediction based on a second learning of the Field-to-Field model, using this time the enriched category descriptions (learning not illustrated here for conciseness and redundancy with frame 1).

other field $\boldsymbol{c_{k'}}$. As an example, when a word present in the *Description* of a job j is related to a *Task* of the category c, its importance differs from the one of a word related to the work *Environment* of c. Similarly, this word has a different importance when present in the *Company Description* of j, or in the job *Title* of j. This reasoning is motivated by the fact that the category description gives a separation of the words into those related to a *Skill*, those related to a *Task*, those related to an *Environment*, an so on.

The first step to describe the Field-to-Field weighting model (FtFw) is to define the FtF matrix between a job advert j and a job category c. This matrix $S(j,c) \in \mathbb{R}+^{n_j \times n_c}$ compares each field of j to each field of c as:

$$S(j,c) = \begin{Bmatrix} S(j,c)_{1,1} & S(j,c)_{1,2} & ... & S(j,c)_{1,n_c} \\ S(j,c)_{2,1} & S(j,c)_{2,2} & ... & S(j,c)_{2,n_c} \\ ... & ... & ... & ... \\ S(j,c)_{n_j-1,1} & S(j,c)_{n_j-1,2} & ... & S(j,c)_{n_j-1,n_c} \\ S(j,c)_{n_j,1} & S(j,c)_{n_j,2} & ... & S(j,c)_{n_j,n_c} \end{Bmatrix} \tag{4.6}$$

where each element $S(j,c)_{i,k}$ represents the normalized similarity between the i-th field of the job j and the k-th textual field of the enriched category c, that is defined as:

$$S(j,c)_{i,k} = \frac{cos(\boldsymbol{j_i}, \boldsymbol{c_k})}{\sum_{p=1}^{n_j} \sum_{q=1}^{n_c} cos(\boldsymbol{j_p}, \boldsymbol{c_q})} \tag{4.7}$$

where the denominator is a normalization factor. By comparing independently each field to another, $S(j,c)$ captures specifically which job field match with which category field. The predicted similarity $sim(j,c) \in \mathbb{R}$ between j and c is then defined as:

$$sim_\lambda(j,c) = \sum_{i}^{n_j} \sum_{k}^{n_c} \lambda_{i,k} S(j,c)_{i,k} \tag{4.8}$$

where $\lambda \in \mathbb{R}^{n_j \times n_c}$ encodes the field-to-field weights. These weights could ideally be manually pre-defined by a exceptionally gifted expert; however in practice we will prefer to compute them using a learning-to-rank approach, as detailed in the next sub-section. Such weights are computed such that the highest $sim_\lambda(j,c)$ is, the most the category c corresponds to the job advert j. However, $sim_\lambda(j,c)$ can not be interpreted as a probability since it is unbounded. The diagram of Figure 4.4 represents visually the FtFw similarity.

The definition of $sim_\lambda(j,c)$ proposes a linear combination of the similarities $S(j,c)_{i,k}$. To justify this, we propose to omit the denominators in Equation (4.7) and in the cosine $cos(\boldsymbol{j_i}, \boldsymbol{c_k})$. These denominators serve for normalization, and without them the expression of $sim_\lambda(j,c)$ (Equation (4.8)) can be re-written as a term-by-term sum:

$$\sum_{i}^{n_j} \sum_{k}^{n_c} \lambda_{i,k} \sum_{t \in \mathcal{V}} w_{\boldsymbol{o_i},t} . w_{\boldsymbol{c_k},t} = \sum_{t \in \mathcal{V}} \sum_{i}^{n_j} w_{\boldsymbol{o_i},t} . \left(\sum_{k}^{n_c} \lambda_{i,k} w_{\boldsymbol{c_k},t} \right)$$

which brings us back to a linear text classification algorithm, the category c being the considered class. Indeed, like in the text classifiers of this type [C.B. Do,

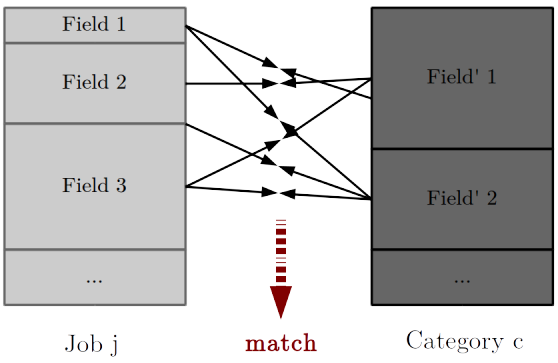


Figure 4.4 – Conceptual schema of FtFw model

2006], each term $t \in \mathcal{V}$ is given a fixed coefficient $\sum_k^{n_c} \lambda_{i,k} w_{\boldsymbol{c_k},t}$, which does not depend on the job advert j, but only on the category c. The specificity of our model is that the coefficients are computed using the category descriptions (represented by $w_{\boldsymbol{c_k}}$), and that coefficients get a boost $\lambda_{i,k}$ depending on the textual field, in order to leverage the documents structure. Also, we use two normalizations for the exact computation of $sim_\lambda(j, c)$: the cosine measure and the denominator of Equation (4.7). They tackle the varying sizes of the fields $\boldsymbol{j_i}$ or $\boldsymbol{c_k}$, which differs from a job to another but also among the fields of a given job advert, and similarly for the categories.

4.3.2 Learning-to-Rank Approach

The efficiency of the FtFw model relies on the weights $\lambda_{i,k}$, for which we propose to adopt a learning-to-rank method with a pointwise approach. In other words, the values for $\lambda_{i,k}$ are computed using a data-set Γ of entries

$$(j, c, y_{j,c}) \in \Gamma$$

Where $y_{j,c} \in \{-1, +1\}$ represents whether c is a correct category or not for the job j. To build Γ, 1,450 job adverts have been randomly selected from Multiposting's database, and for each, few possible match (j, c) (typically 5 per advert) were generated using the top-ranked categories using the preliminary similarity of Equation (4.4). Our experts then manually specified the corresponding $y_{j,c}$ values, considering whether the match was acceptable or not. In practice, in the $8,226$ examples of match job/category of Γ, some jobs match one or multiple categories and others none, and many categories c are simply absent. Consequently, λ is in no way usable for training a multi-class classification algorithm, as the ones described in Section 2.3.1.

Leveraging this data-set, the parameters for $sim_\lambda(j, c)$ can be computed, which requires therefore much fewer annotated jobs than for a classical classifier. Indeed, the variables $\lambda_{i,j}$ have relatively low dimension: $k \times l$ (< 100 in our data-sets), compared to the number of terms in the vocabulary $|\mathcal{V}|$ which is the number of parameters for a textual classifier. We propose to use a SVM, which means we minimize the following quantity:

$$\min_{\lambda,b} \quad \|\lambda\|^2 + \frac{1}{|\Gamma|} \sum_{(j,c,y_{j,c})\in\Gamma} max\left(0,\ 1 - y_{j,c} \times (sim_\lambda(j,c) - b)\right) \tag{4.9}$$

where the first term is the regularization term, and the optimum is found using a gradient ascend. The predicted value for a match is then $sign\left(sim_\lambda(j,c) - b\right)$, but in our system we only use $sim_\lambda(j,c)$ for ranking the categories.

This learning-to-rank method does not necessarily require the linearity assumption of Equation (4.8). Indeed, different combinations of the field-to-field comparisons $S(j,c)_{i,k}$ could be considered, for example a polynom with coefficients learned on the data-set Γ. As another example, a random forest can be trained to separate the matrices $S(j,c)$ in the case of a match or a non-match. This will be vainly experimented later in Section 4.6, confirming the linear assumption.

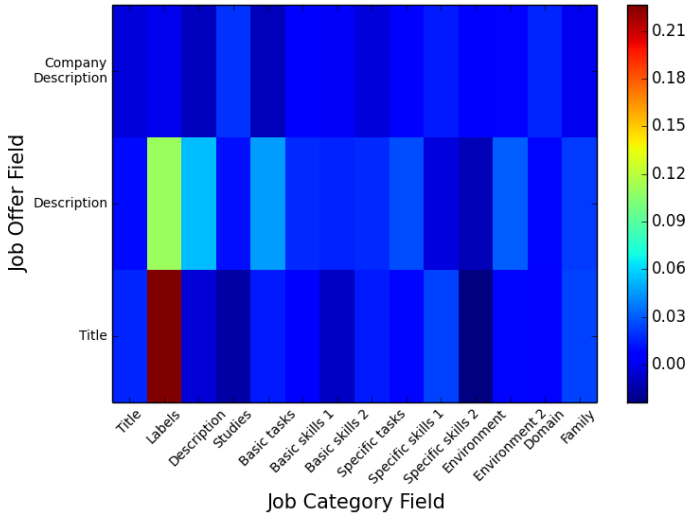


Figure 4.5 – Field to field importances $I_{i,k}$ (Equation ((4.10))). Standard deviation are not displayed, but negligible w.r.t the absolute $I_{i,k}$ values.

The FtFw model can be interpreted by visualizing the computed weights $\lambda_{i,k}$. To do so, we use a bootstrapping approach, meaning we compute the values $\lambda_{i,k}$ on a random half of Γ, for 100 different runs. The learned parameters $\lambda_{i,k}$ are not directly usable for visualization, since the field-to-field comparison $S(j,c)_{i,k}$ in the sum of Equation (4.8) can be structurally very low, meaning the corresponding weight $\lambda_{i,k}$ will be artificially high. In light of this, we instead define the field-to-field importance $I_{i,k} \in \mathbb{R}$ as:

$$I_{i,k} = \lambda_{i,k} \times \frac{1}{|\Gamma|} \sum_{(j,c,y_{j,c})\in\Gamma} S(j,c)_{i,k} \tag{4.10}$$

which is the average value of the term in the sum of equation (4.8) for given i and k values. This corresponds to the similarity between the i-th field and the

k-th field, computed on the set Γ. The importances $I_{i,k}$ are shown in Figure 4.5, with the field names of the job in the y-axis and the fields' names of the job category in the x-axis. We note that the most important job field is the *Title*, as already pointed out by [Ghani, 2002, Diaby et al., 2013]. This is explained by three reasons: it forms a summary of the job advert, it is short and the recruiters are careful when writing it. On the contrary, the job *Description* is long, sometimes noisy, and therefore harder to exploit in automatic language processing. We also note that the field-to-field weights $\lambda_{i,k}$ are almost systematically positive (since $S(j,c)_{i,k} \geq 0$): the presence of common terms in a job advert and a job category has never a negative effect on the match, which is a logical and meaningful result.

4.3.3 Comparison with Basic Field Weighting

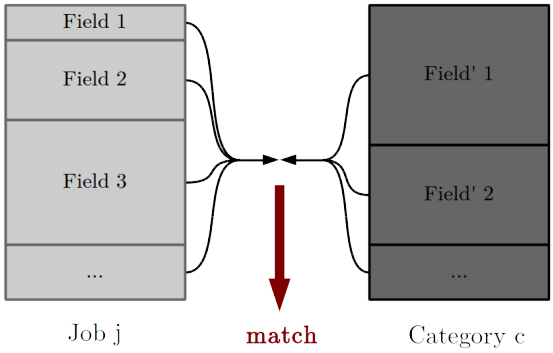


Figure 4.6 – Conceptual schema of models weighting independently the fields.

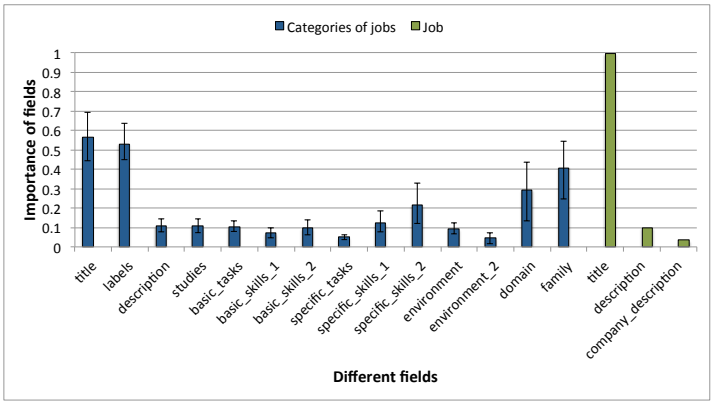


Figure 4.7 – Weights μ_i, ν_k estimated for the fields weighting approach of Equation (4.11)*. Bootstrapping (100 runs) has been used to estimate the* 95% *empirical confidence intervals (*[2.5% *quantile*, 97.5% *quantile*]*).*

A natural discussion about the FtFw model is to compare it with the classical fields-weighting models, commonly used in numerous information retrieval systems [Smiley and Pugh, 2011, Gormley and Tong, 2015]. Conceptually, each field

is given an intrinsic weight, so that the fields are independently aggregated for both documents in the similarity measure, as represented in Figure 4.6. Whereas the weights are manually chosen in [Smiley and Pugh, 2011, Gormley and Tong, 2015], a learning-to-rank approach has been proposed in [Diaby et al., 2013] to compute the optimal weights of each fields. Using the notations of this chapter, in this model the similarity function $sim_{\vec{\mu},\vec{\nu}}(j,c) \in [0,1]$ is defined as:

$$sim_{\vec{\mu},\vec{\nu}}(j,c) = \frac{(\sum_i \mu_i \overrightarrow{w_{j_i}}) \cdot (\sum_k \nu_k \overrightarrow{w_{c_k}})}{\| \sum_i \mu_i \overrightarrow{w_{j_i}} \| . \| \sum_k \nu_k \overrightarrow{w_{c_k}} \|} \tag{4.11}$$

where the weight $\mu_i \in \mathbb{R}$ applies to the i-th field of the job j, and $\nu_k \in \mathbb{R}$ applies to the k-th field of the job category c. The optimal weights are computed by a gradient ascent on the training set Γ, and are visualizable following a similar bootstrapping approach to the previous visualization (Figure 4.7). The job *Title* appears to be again the most important, but we note that the model captures a simpler interaction between the job advert and the job category, since the job *Description* has a low importance in the computation.

4.3.4 Feature Selection for the FtFw Model

The FtFw model naturally leads to a feature selection process [Huan Liu, 2005], which aims to consider only a sub-set of features $S(j,c)_{i,k}$ in the computation. The objective of such process is to reduce computational costs, and possibly improve the precision by removing noise features. We conducted experiments by using a sequential search, more precisely a backward elimination, with an evaluation on the subsets to compare the ROC-AUC [Omary and Mtenzi, 2010]. For a given sub-set of features, we used a 10-fold cross-validation on the data-set Γ, in order to compare the ground-truth values $y_{j,c}$ and the predicted values $sim_\lambda(j,c)$ on this data-set, through computing the ROC-AUC. At each iteration of the process, the number of features is fixed, and the model is successively evaluated for deleting one of the remaining component i,k of $S(j,c)$. Then, the i,k values that leads to the lowest ROC-AUC is deleted for all the following iterations. This means we stop comparing the i-th field of the job to the k-th field of the category, because it does not seem to help much to predict the match.

Figure 4.8 shows the successive AUC-ROC values at each iteration of the feature selection process. One sees that the performance is still very high with only 3 features. To visualize the corresponding selected features, we introduce a feature importance $I'_{i,k} \in \mathbb{R}$, computed as

$$I'_{i,k} = exp\big(\text{deletion order of } S(j,c)_{i,k}\big) \tag{4.12}$$

that are visually comparable with the field-to-field importances $I_{i,k}$ defined in Equation (4.10). Figure 4.9 is indeed very close to the importances shown in Figure 4.5, confirming the intuition that a pair of fields i,k with a high field to field importance $I_{i,k}$ is also very likely to be kept after the feature selection.

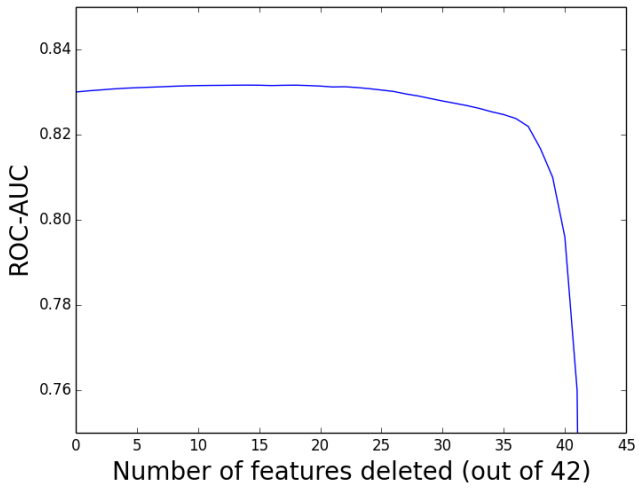


Figure 4.8 – Performance at each step of incremental feature elimination, on Categorization data-set

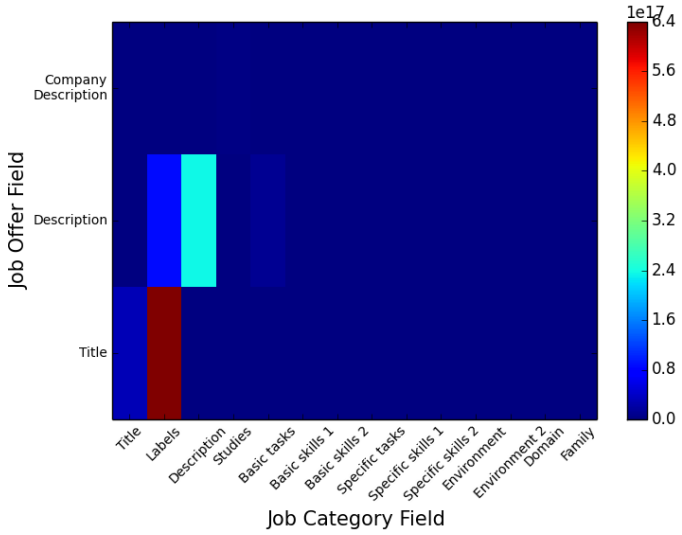


Figure 4.9 – Field to Field importance $I'_{i,k}$ (Equation (4.12)).

As a positive result, this feature selection leads to an interesting reduction of computational costs, in time and in space. Indeed, for this data-set, we only need to vectorize and store vectors for only 2 fields for j and 2 for c, instead of 3 and 14. Moreover, only 3 cosines need to be computed, instead of 42. No large-scale tests have been conducted yet to precisely estimate our computational costs reduction, but significant improvements are expectable using this feature selection.

Despite this computational aspect, the feature selection does not significantly improve the system's precision, measured here through the ROC-AUC. Hence, reducing the problem dimension seems irrelevant for increasing the categorization

precision. Another strategy for improving the system's performance is in a sense the opposite of the feature selection: it is to enrich the features of the problem. One way to do this is to semantically enrich the category descriptions, that we study in the next section.

4.4 Enrichment of Category Descriptions

By computing the job/category similarity using the FtFw model, the job categorization depends purely on the official category descriptions: those documents can therefore be a limit to the system's precision. In light of this, we propose in this section a way to improve these descriptions, using a large data-set of unlabeled job adverts.

4.4.1 Going Beyond the Official Category Descriptions

For the standardization process, using the official descriptions presents some limitations. We will depict this aspect through the example of the description for the "Computer Programming" category. As one can observe in Figure 4.1, the *Skills* field content presents a fixed list of keywords including popular languages (e.g., Java, C++, Python), without including recent languages such as Clojure or Swift that are likely to be in high demand in the immediate future. In the very evolutive context of recruitment, it is indeed considerably hard to update the description and anticipate the rapid changes in the industry - even for a domain expert. These new languages are however present in the job adverts since they directly represent the needs of the job market, for example in the following advert:

Title: Software Engineer Python
Description: You will work with algorithm development. The following Skills and Abilities are required: - Demonstrated knowledge of software development in MySQL / Python / Django Framework - The ability to work unsupervised when required.
Profile: Engineering / Computer Science graduate
Company Description: At ***, the leader in entertainment innovation, science meets art and high tech means more than computer code.

Figure 4.10 – This job advert presents new programming languages, that are absent in the category description of Figure 4.1; it is therefore wrongly categorized when using the FtFw model with the official category descriptions.

A first limit of the official category descriptions is the vocabulary used to write them, represented by the set $\mathcal{V}$. Some terms might indeed be absent from $\mathcal{V}$, for example the new programming languages of the job advert above. More subtly, some terms might be present in $\mathcal{V}$ but under-represented in some category descriptions. In other words, some concepts might not be emphasized enough in a category description, so that the corresponding terms' weights are too low to

properly categorize the related job adverts. In light of this, detecting the dynamic semantic variations is crucial for improving the job categorization, especially in presence of terms in the job advert that could not match with the official descriptions of the profession.

In this section we propose to consider a large corpus $\mathcal{J}$ of job adverts extracted from Multiposting's database. This corpus is assumed to represent the exact needs of the job market and will serve at improving the category descriptions. Each job category description will be enriched with the more contextually relevant terms in the given job corpus, in order to properly reflect the changes we detected in the considered job professions. To do so, our approach bridges the gap between the categories and the documents knowledge in an evolving manner, leveraging the data-set of real-world job adverts that can change over time, making the enrichment is dynamic. From this data-set we associate to each category description a set of enriching keywords, each one being associated to a weight. Such enriching keywords are context and knowledge-aware, so that our enriched descriptions are aware of not only the static knowledge of the domain expert that implemented it, but also the background dynamic knowledge expressed by the real job adverts scenario.

With this objective, the next sub-sections present a novel bottom-up method that aims to automatically enrich each category descriptions with the newest terms detected in the job adverts corpus. This method is articulated in three steps: we firstly associate the context documents to each category, before extracting the relevant keywords from those contexts, and we lastly compute the new keywords' weights used in the FtFw model.

4.4.2 Associating Context Documents to Categories

This first step takes as input a data-set $\mathcal{J}$ of unlabelled job adverts, which are context documents. For our experiments and final implementation we considered the real-world jobs published on Pole Emploi[1] during the year 2014, whose statistics are displayed in table 4.1. This website belongs to the French national employment services, and aims to be the most comprehensive possible about job market. For each job category $c \in \mathcal{C}$, we build a set J_c made up of representative jobs taken from $\mathcal{J}$ and that seem related to c. We capture this notion of relatedness through the similarity $sim_\lambda(j, c)$ computed by the FtFw model. This association, as well as the whole enrichment process, is therefore *semi-supervised*: firstly, we use the manually annotated data-set Γ for computing $sim_\lambda(j, c)$; and secondly, the corpus of unannotated jobs $\mathcal{J}$ serves at improving the similarity $sim_\lambda(j, c)$ by enriching the category descriptions.

For every category c, the set J_c is initially empty and incrementally augmented during the associative process, as follows: for each job $j \in \mathcal{J}$ we

1. Compute its similarity $sim_\lambda(j, c)$ with respect to all the categories using the FtFw model.
2. Consider the top-ranked category $c^* = \underset{c \in \mathcal{C}}{argmax}\ sim_\lambda(j, c)$

[1] `www.pole-emploi.fr`

3. Add j to J_{c^*}

At the end of this phase, each job category $c \in \mathcal{C}$ has a set of job adverts J_c associated to it. Some of those sets can be empty, in the case of rare categories that are not represented in $\mathcal{J}$; in such case the corresponding category will not be enriched, which appears acceptable for rare categories. This step is represented in Bob's world in Figure 4.3, in the second frame. In frame 3, each box - representing a category c - is filled with sheets of papers which correspond formally to the set J_c.

Number of jobs $\lvert\mathcal{J}\rvert$	215,512
Number of categories c enriched, i.e such that $J_c \neq \emptyset$	435
Average number of jobs associated $\lvert J_c\rvert$	4.25
Standard deviation for $\lvert J_c\rvert$	14.25

Table 4.1 – Statistics about the enrichment data-set $\mathcal{J}$ and the set J_c of jobs associated to each category c.

Using the FtFw model described in the previous section, we are now able to associate each job to its best representative category in the given categorization nomenclature. After this step, considering a category c and his document-to-category association, J_c, we aim to search for the most contextual informative keywords for each category.

4.4.3 Extracting Relevant Keywords

This phase proposes to find the keywords better describing the category in the corpus of jobs. Leveraging the associated documents to each category, we can interpret the keywords that are supporting them as the intentions - or the explanations - supporting these categories.

For this purpose, we treat the category c as a query and the extension set J_c as a contextual feedback, and we apply a probabilistic feedback process. The idea is that each job advert j in the subset J_c indicates a positive feedback about the keywords in its aggregated bag of words $\overline{j}$. As discussed in [Ruthven and Lalmas, 2003], the degree of matching between the keyword and the category is then computed by treating each common document between a keyword and a category as a positive relevance feedback; each job corresponding to the keyword but not to the category node is on the contrary treated as a negative relevance feedback. In particular, the relevance weight $rw(c,t)$ for a category c of a candidate term t is theoretically given by the following probabilistic formula:

$$rw(c,t) = log\left(\frac{p(t \in \overline{\boldsymbol{j}} | j \in J_c)}{p(t \in \overline{\boldsymbol{j}} | j \notin J_c)}\right) \left| p(t \in \overline{\boldsymbol{j}} | j \in J_c) - p(t \in \overline{\boldsymbol{j}} | j \notin J_c) \right|$$

Intuitively, the keywords that appear frequently only in specific associations and rarely in others will tend to have higher weights. In other words, the quantity $rw(c,t)$ weighs the most contextually informative keywords for the category c. By re-expressing the probabilities and modifying slightly the above formula

[Ruthven and Lalmas, 2003], the relevance weight $rw(c,t)$ is computed in practice by

$$rw(c,t) = log\left(\frac{\frac{r_t}{R_c - r_{c,t}}}{\frac{n_t - r_{c,t}}{N - n_t - R_c + r_{c,t}}}\right)\left|\frac{r_{c,t}}{R_c} - \frac{n_t - r_{c,t}}{N - R_c}\right| \tag{4.13}$$

Where:

- $N = |\mathcal{J}|$ is the total number of job adverts in the corpus
- $R_c = |J_c|$ is the number of jobs associated to the category c
- $n_t = |\{j \in \mathcal{J} | t \in \overline{\boldsymbol{j}}\}|$ is the total number of job adverts containing the term t.
- $r_{c,t} = |\{j \in J_c | t \in \overline{\boldsymbol{j}}\}|$ is the number of job adverts associated to the category c containing the term t

Leveraging these computed weights, we define the enriching field of the category c, noted $\boldsymbol{c_{l+1}}$, as the multiset of terms t such that $rw(c,t) > 0$. Such terms can be already present in the initial vocabulary $\mathcal{V}$ - obtained from the unenriched category descriptions - but some of them are new, like the ones shown in Figure 4.11. We define the enriched job category as an extended list of multisets of terms $c' = \{\boldsymbol{c_1}, \boldsymbol{c_2}, ..., \boldsymbol{c_{n_c}}, \boldsymbol{c_{n_c+1}}\}$ where $\boldsymbol{c_{n_c+1}}$ encodes the knowledge extracted from the corpus $\mathcal{J}$, and the other fields $\boldsymbol{c_k}$ of the category remain unchanged. By adding the enriching field $\boldsymbol{c_{n_c+1}}$, the vocabulary of our problem is extended, and we note $\mathcal{V}'$ the enriched vocabulary obtained by concatenation of all the fields of the new category descriptions c'.

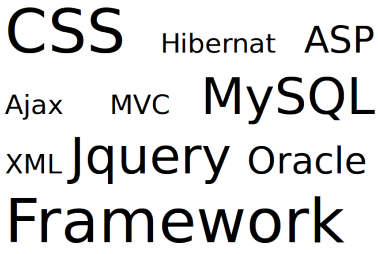


Figure 4.11 – Top ten new terms (with respect to $rw(c,t)$) extracted from $\mathcal{J}$ for the category "Software Engineering".

4.4.4 Final Similarity Measure

Now that the categories are described by an extra enriching field, we will detail how the similarity $sim(j,c')$ between a job j and an enriched category c' is computed. We will again make use of the FtFw model for this, which again involves computing the cosine $cos(\boldsymbol{j_i}, \boldsymbol{c_k})$ and consequently the weights $w_{\boldsymbol{j_i},t}$ and $w_{\boldsymbol{c_k},t}$.

Firstly, the invert document frequency $idf(t)$ is updated by being computed on the enriched category descriptions $\overline{c}'$, instead of on the initial ones $\overline{c}$. If the value change will be low for the already considered terms $t \in \mathcal{V}$, the updated $idt(t)$ will be now non-null for the new terms of $\mathcal{V}'$. This new value for $idf(t)$ is used for weighting the terms in the initial categories' fields $\boldsymbol{c_k}$ as well as in the job fields $\boldsymbol{j_i}$ (Equation (4.3)). Secondly, for the new field-to-field comparisons $cos(\boldsymbol{o_i}, \boldsymbol{c_{l+1}})$ with the enriching field $\boldsymbol{c_{l+1}}$, we will use a novel term weighting function $w_{\boldsymbol{c_{l+1}},t}$. Indeed, simply updating $idf(t)$ would mean we do not make profit from the relevance weights $rw(c,t)$. This quantity gives however a significant clue about the importance of the term t. We propose therefore to calculate the term weight $w_{\boldsymbol{c_{n_c+1}},t}$ of the term t for the category c as:

$$w_{\boldsymbol{c_{n_c+1}},t} = \sqrt{tf(c,t)} \left(1 + log\big(idf\,(t)\,\big) + rw(c,t)\right) \tag{4.14}$$

where $rw(c,x)$ is the relevance weight and $tf(c,t)$ is the number of jobs associated to the category c. The homogeneity of the weighting function is assured by expressing $rw(c,t)$ as a log of probabilities, since $idf(t)$ is assimilated to the probability of observing term t in a category [Ruthven and Lalmas, 2003].

By using this weighting function when computing the similarities with the enriching field $cos(\boldsymbol{j_i}, \boldsymbol{c_{n_c+1}})$, the FtF matrix $S(j,c')$ for the enriched category c' is augmented of one column compared to $S(j,c)$, which corresponds to $\boldsymbol{c_{n_c+1}}$. Consequently, the sum of similarities in Equation (4.8) includes these new terms when computing $sim_\lambda(j,c')$, so that the parameter λ of the model is also augmented. By a slight abuse of notations, $sim_\lambda(j,c')$ will denote this new similarity, but $\lambda \in \mathbb{R}^{n_j \times (n_c+1)}$ has a different shape than when writing $sim_\lambda(j,c)$. This parameter is again optimized through a second training on Γ, meaning that the annotated data-set serves twice: firstly, for the training of the FtFw model with initial categories, used for assigning jobs to the categories during the enrichment; secondly, for the FtFw model with the enriched categories. This double use of the data-set will be handled subtly during the evaluation process.

After the enrichment process, given a job advert j to be standardized, the similarities $sim_\lambda(j,c')$ are computed and the top-ranked job category is assigned to the advert j. This final assignment of job category is quite simple, but efficient in practice, as experiments of Section 4.6 will show. However, this decision does not provide any precise idea on how correct the automatic categorization is. Indeed, given a job advert, the score $sim_\lambda(j,c')$ has no probabilistic interpretation, and is only meaningful when compared to other categories scores. To tackle this, the next section proposes a way to estimate the probability that the system's answer is correct.

4.5 Probability Estimate for Categorization

In this section, we propose to estimate the probability that the predicted category for a given job is correct, based on the similarity scores produced for the ranking. We propose and evaluate different estimations for this quantity, defined as correctness. This work extends to all learning-to-rank system with a pointwise

approach, as it was successfully experimented in [Malherbe et al., 2015c] for the schema mapping of Chapter 3.

4.5.1 Definition of Correctness

As for any predicting system, it would be relevant to estimate a probability for each prediction of job category. The probability estimate is a recurrent problem of supervised learning, as detailed in Section 2.3.1, and having a probabilistic prediction is for instance used in semi-supervised learning and active learning, that both require insight about the quality of the prediction. In the case of job categorization, a probability estimate is also useful to filter the poorly categorized jobs of a data-set before computing some statistics.

In our approach, after the enrichment of the category descriptions, the job categorization consists in ranking the categories c with respect to a job advert j using $sim_\lambda(j, c')$. To investigate the accuracy of the predicted category, we propose to step back from our problem, and consider *only* the ranking scores produced by the system. Following such hypothesis, our following study applies to each of the similarity scores we presented above, namely $sim(j, c)$, $sim_\lambda(j, c)$, $sim_{\vec{\mu},\vec{\nu}}(j, c)$ defined by Equations (4.4), (4.8) and (4.11). For the experiments, we will focus on the final similarity score $sim_\lambda(j, c')$ (computed after enrichment), but for clarity and generality, we will simply write $sim(j, c)$ in the formulas.

Here are few points worth noticing about the similarity score $sim(j, c)$. Firstly, it should ideally represent the quantity $p(y_{j,c} = 1|j, c)$, but has no probabilistic interpretation, since it is unbounded, with $sim(j, c) \in \mathbb{R}^+$. Secondly, the function sim is computed *independently for each category* c, meaning this is a learning-to-rank model with a pointwise approach. Thirdly, the function sim results from a supervised (or semi-supervised) learning, meaning the values it produces depend on the data-set used for the training, namely Γ.

A specific aspect of our standardization problem is that one and only one correct category c is needed for a given job advert j. In other words, once a relevant category is found, the other ones are useless, and the ranking only serves at finding the most likely category c^*, that is to say the top-ranked one:

$$c^* = \underset{c \in \mathcal{C}}{argmax}\ sim(j, c)$$

And in this context, we want to determine how likely the category c^* is to be correct given a job advert j. We define therefore correctness C_j as the probability that the top-ranked category is a valid for j:

$$C_j = p\left(y_{j,c^*} = 1\right) = p\left(c^* \text{ is a correct category for } j\right)$$

This definition is generic, as it applies to any ranking system for which we systematically select the top-ranked solution. We need to clarify the difference between correctness C_j and the highest score $sim(j, c^*)$: first, C_j is a probability, contrary to this score. Second, scores are computed with independence assumption (as illustrated in Figure 4.12), whereas C_j takes into account that c^* is the top ranked category, and is implicitly aware of the other categories. For instance,

even if all the scores $sim(j, c)$ are very low but $sim(j, c^*)$ remains much higher than other scores, c^* is likely to be a correct answer. On the other hand, if every category has a high score but no outcome seems to stand out from the others, we are less confident about c^* being a correct category.

Our objective is to *estimate correctness C_j the most precisely as possible, based on the scores $sim(j, c)$*. To do so, the data-set Γ provides crucial information: for each job j represented in Γ, we can compute the corresponding scores $sim(j, c)$ and the top category c^*. When (j, c^*, y_{j,c^*}) is present in Γ, we have an example of truly or wrongly categorized job advert:

$$(j, y_{j,c^*}) \in \Upsilon$$

where Υ is a data-set that will serve for correctness estimate, and is obtained by re-expressing Γ for a given a ranking measure $sim(j, c)$. This second data-set Υ is smaller than Γ, since there is only one entry per job j, and j is present in Υ only if the match with the top-ranked category y_{j,c^*} is given by Γ. One notes that Υ changes with the similarity function $sim(j, c)$, since the top-ranked category c^* changes. In the following, we will see how to estimate correctness by leveraging the scores $sim(j, c)$, with and without the feedback provided by the data-set Υ.

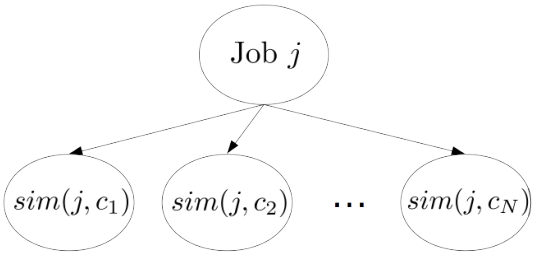


Figure 4.12 – Bayesian network representing the independence assumption among scores $sim(j, c)$. This property is verified by any learning-to-rank system with a pointwise approach.

4.5.2 Various Approaches for Estimating Correctness

We propose here different approaches for estimating correctness C_j, that introduce incrementally our final approach based on the top-k scores.

Heuristic-Based Approach

A first natural approach is to rely directly on the scores $sim(j, c)$ predicted for the ranking. Given a job j, in order to interpret each score as the probability that the category c is valid for j, we normalize the scores by

$$sim'(j, c) = \frac{sim(j, c)}{\sum_c sim(j, c)}$$

in order to have $sim'(j,c) \in [0,1]$ and $\sum_c sim'(j,c) = 1$. We use then the following metrics proposed by [Vanrompay and Berbers, 2012] as estimations for correctness:

- *Maximum* equals the highest normalized score: $\hat{C}_j = \max_c sim'(j,c) = sim'(j,c^*)$
- *Distance* evaluates how much the best category stands out from the second one: $\hat{C}_j = \max_c sim'(j,c) - 2^{nd}\ \max_c sim'(j,c)$

We note that these heuristic-based approaches rely on only the highest (for the Maximum) and the two highest values for $sim'(j,c)$ (for the Distance), whereas more values could be considered. Such approaches are moreover *static*, since their computations do not exploit the categorization examples of Υ. We tackle this second aspect in the following approach.

Learning on Independent Scores

A second approach is to use Γ as a training data-set. For a given job advert j, we consider the score $sim(j,c)$ for a single category c and forget the scores for other categories of $\mathcal{C}$. It makes sense to separate every category since the ranking is computed independently among categories. We try thereby to estimate the probability that a category c is valid for j just from the score $sim(j,c)$ - that is to say $p\big(y_{j,c} = 1|sim(j,c)\big)$ - through a predicting function ψ:

$$p\big(y_{j,c} = 1|sim(j,c)\big) = \psi\big(sim(j,c)\big) \tag{4.15}$$

where $\psi : x \in \mathbb{R} \to [0,1]$ is learned on a set of entries $\big(sim(j,c),\ y_{j,c}\big)$ obtained from the data-set Γ (see Section 4.5.3 for some examples for ψ). This function ψ serves at re-expressing the scores $sim(j,c)$ as probabilities, and would therfore be useless if the scores were probabilistic. Contrary to the multi-class probability estimation, this prediction does not depend on the category c, but just on the ranking score, so that c does not need to be represented in Γ for estimating $p\big(y_{j,c} = 1|sim(j,c)\big)$. The estimated correctness C_j for the job j is then obtained by converting the top score into a probability:

$$C_j = \psi\big(sim(j,c^*)\big) = p\big(y_{j,c^*} = 1|sim(j,c^*)\big) \tag{4.16}$$

We note that this final prediction of C_j depends only on the category with the highest value c^*. We expect however that the scores for other categories give additional information on the ranking quality: indeed, if the second highest score $sim(j,c)$ is much lower than the highest one, it means that the first category stands out of the others. In light of this, we propose in the following a model that considers not one but several scores when estimating C_j.

Learning on Top-k Scores

Given a job advert j, we extend the previous learning by considering all the categories scores $sim(j,c)$, meaning we estimate correctness C_j by $p\big(y_{j,c^*} =$

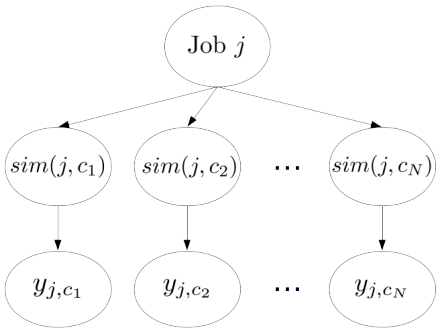


Figure 4.13 – Graphical model for estimation on independent scores.

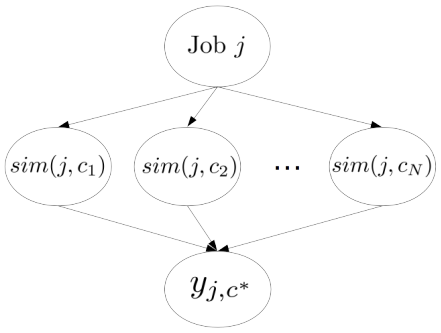


Figure 4.14 – Graphical model for estimation on top-k scores.

$1|sim(j,c)\ \forall c \in \mathcal{C}\big)$. However, the input features can not simply be all the scores $sim(j,c),\ c \in \mathcal{C}$ as is. Indeed, the number of features $|\mathcal{C}|$ would be high and we might encounter overfitting. Besides that, only few categories c are represented in Γ, so that we can not use a fixed order for representing the scores $sim(j,c)$ in an input vector: this would penalize the absent categories in Γ. For the same reason, the use of the principal components analysis would also penalize some categories. We therefore build a vector by re-ordering the scores $sim(j,c)$. To do so, we consider the ranked k highest values of $sim(j,c)$, where $k \in \mathbb{N}$. One can link this process with the distance D computation, where the 2 highest components are considered, statically. We write the top-k scores vector $\overrightarrow{top_k}(j)$:

$$\overrightarrow{top_k}(j) = \begin{pmatrix} \max\limits_{c \in \mathcal{C}} sim(j,c) \\ 2^{nd} \max\limits_{c \in \mathcal{C}} sim(j,c) \\ \vdots \\ k^{th} \max\limits_{c \in \mathcal{C}} sim(j,c) \end{pmatrix} \in \mathbb{R}^k$$

which can be viewed as a *feature map* of a job j into $\mathbb{R}^k$. It captures the distribution of scores, and in particular, how much the first category c^* stands out from the others. The correctness C_j is then estimated by:

$$C_j = \varphi\big(\overrightarrow{top_k}(j)\big) = p\big(y_{j,c^*} = 1 | k \text{ top scores } sim(j,c)\big) \tag{4.17}$$

Where the prediction function $\varphi : \vec{X} \in \mathbb{R}^k \to [0,1]$ is learned on a set of entries $\big(\overrightarrow{top_k}(j),\ y_{j,c^*}\big)$, by re-expressing Υ. The choice for function φ is discussed in the following.

4.5.3 Experimental Precision of Approaches

For our experiments on correctness estimation, we used a two levels cross validation: a first level separates the training and testing sets used for learning ψ and φ, and another cross validation is performed on each training set to produce *unbiased* values $sim(j,c)$. Indeed, when we re-express the data-set Υ as entries $\big(sim(j,c),\ y_{j,c}\big)$ and $\big(\overrightarrow{top_k}(j),\ y_{j,c^*}\big)$, the scores $sim(j,c)$ depend on the data-set

used for training; the objective is to learn the functions ψ and φ from scores $sim(j,c)$ and vectors $\overrightarrow{top_k}(j)$ that reflect what would be predicted for an new job advert j.

Since the approaches on independent scores $sim(j,c)$ and top-k scores $\overrightarrow{top_k}(j)$ involve learning functions ψ and φ to estimate correctness C_j, we compared the following supervised algorithms, whose output can be interpreted as a probability ([Carrizosa and Morales, 2013]):

Sigmoid: $\psi(\vec{x}) = \frac{1}{1+e^{-\vec{\beta}.\vec{x}+\alpha}}$, where $\alpha \in \mathbb{R}$ and $\vec{\beta}$ has the dimension of $\vec{x}$.

K nearest neighbors (kNN): $\psi(\vec{x}) = \frac{1}{n}\sum_i^n \frac{y_i}{|\vec{x}-\vec{x}_i|}$ is a weighted average of the n closest vectors $\vec{x}_i$ in the base ($n = 20$ in our experiments).

Random forest: $\psi(\vec{x}) = \frac{\sum_{trees} P_{tree}(y=1|\vec{x})}{|\{trees\}|}$ is the average of the probabilities $P_{tree}(y = 1|\vec{x})$ produced by each randomized tree in the forest (300 in our experiments).

To estimate the quality of our predictions, for each entry $(j, y_{j,c^*}) \in \Upsilon$ we compared the estimated correctness C_j to the ideal prediction, y_{j,c^*}, using 3 metrics: the ROC-AUC as defined in [Carrizosa and Morales, 2013], the mean square error MSE and the log-likelihood LL:

$$MSE = \frac{1}{|\Upsilon|}\sqrt{\sum_{(j,y_{j,c^*})\in\Upsilon}(C_j - y_{j,c^*})^2} \qquad LL = \frac{1}{|\Upsilon|}\sum_{(j,y_{j,c^*})\in\Upsilon} log(|C_j - y_{j,c^*}|)$$

		ROC-AUC	LL	MSE
Heuristic Based	Maximum	0.73 ± 0.03	-1.41 ± 0.14	0.49 ± 0.05
	Distance	**0.75 ± 0.00**	-2.20 ± 0.23	0.57 ± 0.06
Independent Scores (Equation (4.16))	kNN	0.70 ± 0.03	-4.02 ± 1.82	0.26 ± 0.02
	Forest	0.70 ± 0.03	-12.56 ± 3.67	0.28 ± 0.02
	Sigmoid	**0.76 ± 0.02**	**-0.72 ± 0.02**	**0.22 ± 0.00**
Top-k Scores (Equation (4.17))	kNN	0.76 ± 0.02	1.41 ± 0.56	**0.19±0.01**
	Forest	0.74 ± 0.02	-0.58 ± 0.03	0.20 ± 0.02
	Sigmoid	**0.77 ± 0.02**	**-0.58 ± 0.01**	0.20 ± 0.01

Table 4.2 – Quantitative comparison of correctness estimates.

Table 4.2 shows the quantitative results. On the top part, we see that heuristic-based approaches seem to detect well the correct categorizations according to the ROC-AUC, but have a poor probabilistic interpretation, as the low LL and MSE values point out. The estimation from independent scores produces a better probability, but the ROC-AUC remains quite low, suggesting not to consider the scores $sim(j,c)$ independently for our problem. For the estimate on top-k scores $\overrightarrow{top_k}(j)$, the metrics have been computed for $k = 4$, following the curves of the Figure 4.15. Like for the estimation on independent scores, using the Sigmoid for φ gives a high ROC-AUC and approaches relatively well the ideal probability (see the LL values), so that this function fitted on top-k scores gives the best results.

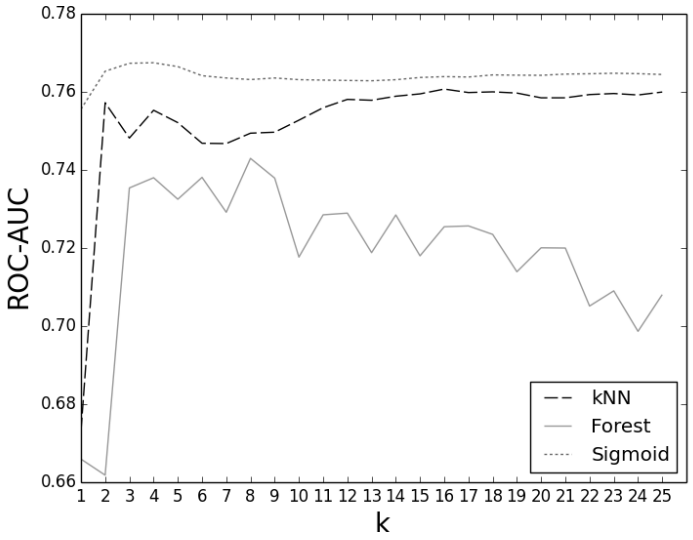


Figure 4.15 – AUC-ROC with respect to k when estimating using the top-k scores. Some overfitting occurs for $k > 4$, so that $k = 4$ seems to be the best value for our final correctness estimate.

4.5.4 Direct Applications of Correctness

The direct application of estimating C_j is to filter the better categorized jobs, by applying a threshold $\varepsilon \in [0, 1]$ on C_j. This serves in practice at computing category-based statistics: for instance, in the SmartSearch project, we want to compute statistics for a given pool of jobs related to a given a category c. By restricting this pool to jobs j such that $C_j > \varepsilon$, the statistics are less biased by the wrongly categorized job adverts, and therefore less noisy. Below here is a study of how to choose the threshold ε.

We applied a threshold ε to the correctness of each job represented in Υ, in order to detach a sub-part with better categorized jobs. Let $Coverage_\varepsilon$ be the proportion of jobs j from Υ such that $C_j > \varepsilon$, and $Accuracy_\varepsilon$ the categorization accuracy on this subpart. We plotted the accuracy with respect to the coverage in Figure 4.16, each point corresponding implicitly to a threshold. The curves show which correctness estimate permits to detach efficiently some better categorized jobs: if we want a sub-part with 90% of jobs correctly categorized, we cover 31% of cases when C_j is estimated using top-k scores, against 25% using independent scores and 19% using distance. Furthermore, the curve is more regular with the estimation learned on top-k scores, which is preferable for industrial strategy. When focusing on a single model, plotting the accuracy/coverage curve serves at defining the industrial strategy through the choice of ε.

Another use of correctness is for ranking the job adverts for a given category. In SmartSearch's interface, the user might have a look at the pool of job adverts used for the statistics. This pool sometimes present poorly categorized job adverts, which is very bad for the user experience; by ranking the jobs j by a decreasing correctness C_j, we ensure that the first displayed adverts most likely belong to the given category. In practice, for a vast majority of the categories,

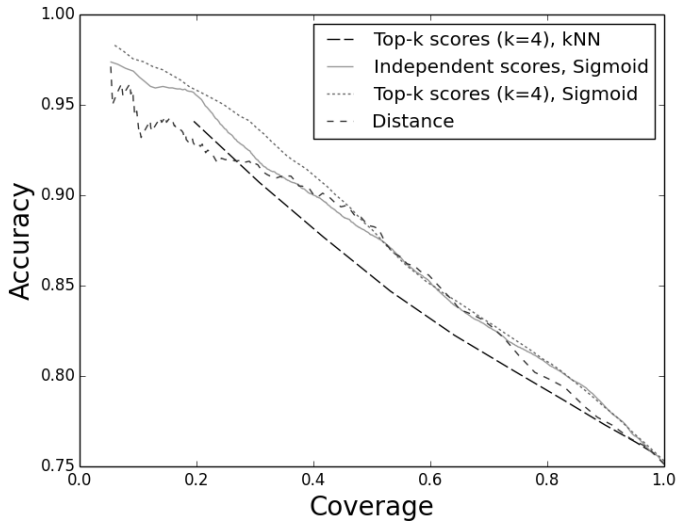


Figure 4.16 – $Accuracy_\varepsilon$ with respect to the $Coverage_\varepsilon$, when applying a threshold ε on correctness C_j (each point corresponds to a threshold value).

there are a large number of associated job adverts in SmartSearch database, at least few hundreds, meaning that the first wrongly categorized job adverts appears at a high rank when using this correctness-based ranking.

Estimating correctness finds also application in co-training. Indeed, this semi-supervised method requires to estimate which jobs of an unannotated corpus are the most correctly categorized, as detailed in the next section.

4.6 Evaluation of Job Categorization

In this section, we evaluate our job categorizer, that is semi-supervised since it leverages the annotated data-set Γ as well as the corpus of non-annotated jobs $\mathcal{J}$, as illustrated in Figure 4.17. For this reason, we propose to compare our system with a co-training on Γ and $\mathcal{J}$, as detailed below.

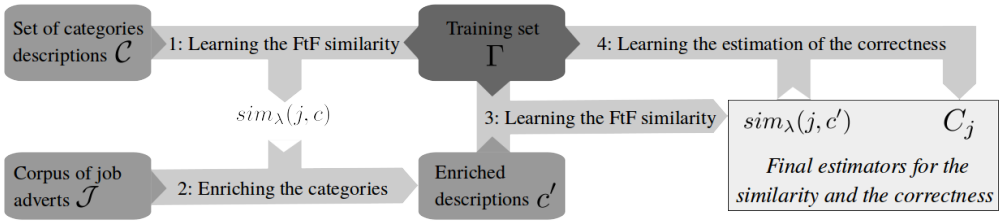


Figure 4.17 – Diagram describing the data-set and the steps - numerated in order of construction - involved for the final job categorizer.

4.6.1 Co-Training for our Learning-to-Rank System

Our system being semi-supervised, a natural comparison is to the co-training. As stated in Section 2.3.2, the idea behind it is to incrementally learn two independent and complementary predictors, which serve at annotating the jobs $\mathcal{J}$ and enriching therefore the training set Γ. It has been successfully applied to job adverts in a multi-class classification [Ghani, 2002], by separating the classification into one based on the title and one on the description. The former textual feature is written $\boldsymbol{j_1}$ in this chapter, while the latter is written $\boldsymbol{j_2}$. This does not directly apply to our system, that adopts learning-to-rank approach, for which co-training has been successfully applied [Tan et al., 2004, Wang and Li, 2011] where the learning is pairwise. To fit our system - that is pointwise - and leverage the correctness estimation, we propose a novel co-training strategy.

With the objective of separating the FtFw model into two independent rankers, we learn on one side $sim_\lambda^{(1)}(j,c)$ based the title $\boldsymbol{j_1}$ on one side, and on the other side $sim_\lambda^{(2,3,4)}(j,c)$ based on the job fields $\boldsymbol{j_2}, \boldsymbol{j_3}, \boldsymbol{j_4}$. This means we consider the first row of the FtF matrix $S(j,c)$ which correspond to the title of the job (Equations (4.7) and (4.8)) for training $sim_\lambda^{(1)}(j,c)$, and the other rows for training $sim_\lambda^{(2,3,4)}(j,c)$, by leveraging initially the data-set Γ. These two rankers will serve at forming incrementally an enriched training set, that we write Γ', using the corpus $\mathcal{J}$. To assess which job advert j of $\mathcal{J}$ is the most correctly categorized by one of the two rankers, we propose to use the correctness - written $C_j^{(1)}$ for the system using $sim_\lambda^{(1)}(j,c)$, and $C_j^{(2,3,4)}$ for the one using $sim_\lambda^{(2,3,4)}(j,c)$. For comparison purpose, we use the non-enriched category descriptions in the similarity measures.

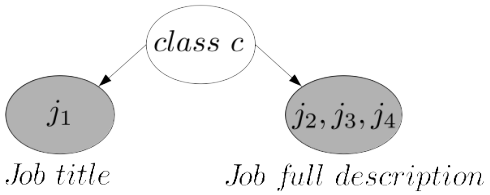


Figure 4.18 – Graphical model representing the assumption behind the co-training. This corresponds to Figure 2.8 of Section 2.3.2, re-expressed with out notations in the case of jobs.

Before detailing the co-training method we propose, let us discuss about the assumptions behind the separation of our system into two rankers $sim_\lambda^{(1)}(j,c)$ and $sim_\lambda^{(2,3,4)}(j,c)$. For the co-training to work properly, the jobs adverts need to reflect the bayesian assumption of Figure 4.18, that gives

$$\begin{aligned} p(j,c) &= p(c)p(\boldsymbol{j_1}|c)p(\boldsymbol{j_2},\boldsymbol{j_3},\boldsymbol{j_4}|c) \\ &= \frac{1}{p(c)}p(c|\boldsymbol{j_1})p(\boldsymbol{j_1})p(c|\boldsymbol{j_2},\boldsymbol{j_3},\boldsymbol{j_4})p(\boldsymbol{j_2},\boldsymbol{j_3},\boldsymbol{j_4}) \\ &\propto p(c|\boldsymbol{j_1}) \times p(c|\boldsymbol{j_2},\boldsymbol{j_3},\boldsymbol{j_4}) \end{aligned}$$

when assuming that the categories c are equiprobable in the job market, and noticing that for a fixed job advert j, $p(\boldsymbol{j_1})$ and $p(\boldsymbol{j_2}, \boldsymbol{j_3}, \boldsymbol{j_4})$ remain constant. One can estimate the two probabilities of the last expression using the approach of Section 4.5.2 (Equation (4.15)), by training sigmoids $\psi_{(1)}$ and $\psi_{(2,3,4)}$ on the independent scores. The former sigmoid is based on the scores $sim_\lambda^{(1)}(j, c)$, while the latter is based on the scores $sim_\lambda^{(2,3,4)}(j, c)$. These estimates give the following similarity function $sim_{bi}(j, c)$ between a job j and a category c:

$$sim_{bi}(j, c) = \psi_{(1)}\big(sim_\lambda^{(1)}(j, c)\big) \times \psi_{(2,3,4)}\big(sim_\lambda^{(2,3,4)}(j, c)\big) \tag{4.18}$$

This variant of the similarity function leverages the FtFw model on non-enriched category descriptions as well as the probability estimate on independent scores. We will not use it for comparison purpose, but for justifying the use of co-training with the two co-rankers $sim_\lambda^{(1)}(j, c)$ and $sim_\lambda^{(2,3,4)}(j, c)$.

The co-training process we propose for our experiments incrementally builds the data-set Γ' by iterating: while all job adverts in the corpus $\mathcal{J}$ are not represented in Γ', do

1. Train on $\Gamma \cup \Gamma'$ the FtF similarity $sim^{(1)}(j, c)$ based on $\boldsymbol{j_1}$, and the similarity $sim^{(2,3,4)}(j, c)$ based on $\boldsymbol{j_2}, \boldsymbol{j_3}, \boldsymbol{j_4}$. Deduce then the respective estimators for the correctness $C_j^{(1)}$ and $C_j^{(2,3,4)}$ after a cross-validation on the initial training set Γ, as explained in Section 4.5.3.

2. Consider the job $j \in \mathcal{J}$ not yet represented in Γ' with the highest correctness $C_j^{(1)}$. Add the corresponding positive example $(j, c^*, +1)$ to Γ'. Since we need negative examples, we also add n_{neg} random negative examples $(j, c, -1)$ to Γ'. This is a negative random sampling [Cochran, 1953] with the same job j paired with n_{neg} random categories c ($c \neq c^*$).

3. Consider $j \in \mathcal{J}$ not yet represented in Γ' with the highest correctness $C_j^{(2,3,4)}$. Add a positive example in Γ', and n_{neg} negative examples like in the previous step.

In order to reduce computational costs, the training of the first step does not need to be performed at each iteration; in our experiments, we updated the similarities $sim^{(1)}(j, c)$ and $sim^{(1)}(j, c)$ at every batch of $1,000$ jobs of $\mathcal{J}$ considered for increasing Γ'. This process differs from the official co-training for multiclass classification, in which the most confident examples are added *for each class*, which implicitly creates negative examples for all other classes. In our case, considering the most confident for each class does not make sense, as each class is not represented in $\mathcal{J}$. Actually, since we build a data-set Γ with binary validation, we are closer to a binary co-training, in which for each most confident example added as true, n_{neg} less confident examples are added in false. But in our case, the most confident example is estimated through the correctness, and the less confident ones result from a random sampling. Due to the core role of the correctness in this process, we prefer to use Γ when learning to estimate it, and keep Γ' for training $sim^{(1)}$ and $sim^{(2,3,4)}$.

At the end of the co-training process, the data-set $\Gamma \cup \Gamma'$ is used for training the FtFw model $sim_\lambda(j, c)$, using non-enriched category descriptions for comparison purposes. The co-training uses the corpus of unannotated job adverts $\mathcal{J}$ in a significantly different strategy than ours: while the section 4.4 aims to enrich the category descriptions $c \in \mathcal{C}$, the co-training aims to enrich the training set Γ used for learning the weights λ used in the FtFw model. In other words, for the standardization problem, the categories enrichment serves at improving the nomenclature $\mathcal{N}$ while the co-training serves at improving the standardization function $f(\boldsymbol{d})$.

4.6.2 Experimental Setup

In order to validate the system we propose for categorizing a job advert j, we dissected our model into different variants, to highlight the improvement of each contribution, namely the FtFw model and the category descriptions enrichment. We also confronted our system with several baselines, so that we evaluated the following methods:

Cosine Measure CM: as a first baseline, the cosine similarity $sim(j, c)$ is the simplest approach, with no learning nor enrichment (Equation (4.4), Section 4.2.2).

Field Weighting Fw: a second baseline is the fields' weighting, whose similarity $sim_{\vec{\mu},\vec{\nu}}(j, c)$ is trained on Γ (Equation (4.11), section 4.3.3).

FtFw: to evaluate the FtFw model, we computed the similarity $sim_\lambda(j, c)$ with non-enriched category descriptions (Equation (4.8), section 4.3.1).

FtFw with Random Forest: to validate the linearity assumption of the FtFw model, we computed a random forest [Breiman, 2001] based on the FtF matrix $S(j, c)$ (4.7) (100 trees in our experiments). The matrix is flattened and considered as an input vector.

Co-training: A baseline of semi-supervised learning is the co-training described in the previous sub-section, with the FtFw model $sim_\lambda(j, c)$ trained on the resulting data-set $\Gamma \cup \Gamma'$ with non-enriched categories. We tested it with different values for the random negative sampling, $n_{neg} = 1$, 5 and 30, and due to the randomness we performed this process 3 times for each n_{neg} value.

Bi-ranking: to validate the assumption behind the co-training, the similarity $sim_{bi}(j, c)$ was computed with non-enriched categories (Equation (4.18)), to assess how independent and complimentary the two sub-features $\boldsymbol{j_1}$ and $(\boldsymbol{j_2}, \boldsymbol{j_3}, \boldsymbol{j_4})$ are.

e-FtFw: this final model leverages the FtFw model $sim_\lambda(j, c')$ when computed with the enriched category descriptions (Section 4.4.4).

Apart from the first model (CM), we used of the training set Γ requires to perform a cross-validation, that was 5-fold in our experiments. A special evaluation process is necessary for the Bi-ranking and Co-training systems. Indeed, the former requires to estimate the probability $p\big(y_{j,c} = 1|sim(j,c)\big)$ while the latter requires to estimate the correctness C_j, through a supervised learning. As explained in Section 4.5.3, for both systems, a second level cross-validation is necessary to train the corresponding correctness estimators. For the e-FtFw, every fold gives different enriched category descriptions, since the similarity $sim_\lambda(j,c)$ differs at each fold, and is used when constructing the associative sets J_c from which the new keywords for the category c are extracted. Similarly, in the Co-training method, at every fold is produced a different enriched training set Γ'. At each fold of the cross-validation, we computed the following metrics on the test fold:

Categorization error rates E_1, E_2: As the top category is particularly important, we first observed the predictions for the top ranked category c^*. The error rate $E_1 \in [0,1]$ is calculated as the proportion of mis-categorized jobs in the data-set Υ. The second rate E_2 corresponds to the industrial needs, and is less strict than E_1: it evaluates if the category proposed by each approach falls in the extended set of valid categories. This set includes the correct category as well as the categories similar to it, that are specified in the category description for the ROME nomenclature. One notes that E_1 and E_2 will be computed on a slightly different number of jobs j depending on the model: indeed, when j, c^*, y_{j,c^*} is or not in Γ, we do not know the ground truth for c^*. Despite this aspect, these errors express the best the real use and industrial needs of the system.

Matching error rate E_{sim}: This second rate evaluates the matching errors with respect to the ground truth, when predicting $y_{j,c}$ based on the similarity between j and c. This rate is computed for all pairs represented in Γ, and is fundamentally different from the multi-class categorization error E_1, although related. As we will see, when the pairwise match is better predicted, the categorization precision also increases.

Receiver Operating Characteristic *AUC-ROC* [Omary and Mtenzi, 2010]: This last measure also evaluates the pairwise matching error on Γ, but aims to be independent from the threshold applied on the similarity measure before the conversion in binary match. It is computed by plotting the fraction of True Positives with respect to the fraction of False Positives (ROC curve), in order to measure the area under the curve (AUC). Contrary to the previous metrics, the higher the $ROC - AUC$ is, the better the pairwise match are predicted.

These measures were computed for each variant of the job categorization to assess the quality of the predictions.

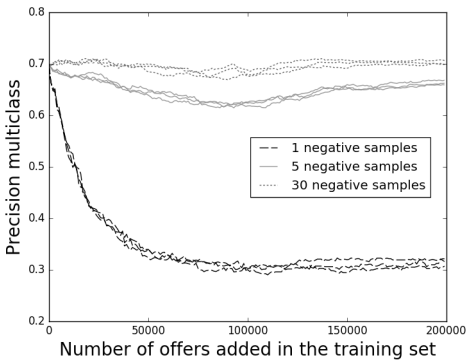


Figure 4.19 – Evolution of the multi-class precision during the co-training process, for $n_{neg} = 1, 5$ and 30.

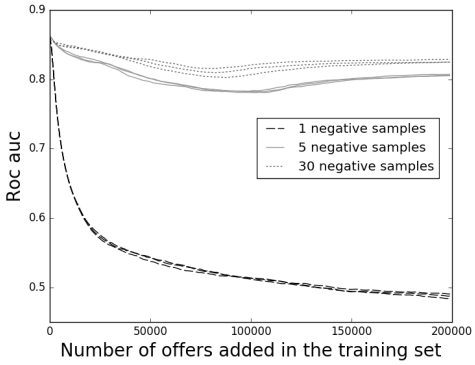


Figure 4.20 – Evolution of the $ROC - AUC$ during the co-training process, for $n_{neg} = 1, 5$ and 30.

4.6.3 Results

For studying the semi-supervised approach, we first evaluated the co-training with 3 values for n_{neg}, through 3 different runs for each in order to see the effect of the randomness in the negative sampling. We displayed in Figures 4.19 and 4.20 the evolution of E_1 and $ROC - AUC$ through the iterations, by learning the FtFw similarity $sim_\lambda(j, c)$ on the partially enriched data-set $\Gamma \cup \Gamma'$. We see that sampling a higher number of negative examples gives better results, and that the randomness has a limited effect on the co-training process, the 3 runs giving very similar results for each n_{neg}. We also observe from the $ROC - AUC$ values that the generated data-set Γ' contains meaningful examples for $n_{neg} = 30$, at the number of $31 \times 215,512$. Nevertheless, this does not improve standardization, contrary to the category descriptions enrichment as detailed below.

Table 4.3 shows the result of the models comparison, with the averaged metrics for $n_{neg} = 30$ for the *Co-training* approach. We first note that the *CM* model gives the worst results, but the values remain in an acceptable range, confirming the approach of matching the job category descriptions c to the job advert j. When compared to this baseline, the matching models *Fw*, *FtFw* and *FtFw with Forest* all improve the metrics, confirming the relevancy to leverage the data-set Γ. Furthermore, the *FtFw* model stands out from the two others, validating therefore the idea to compare each field content of j with each field content of c, as well as the linear combination of those comparisons.

The three last models concern the enrichment process, and the best strategy appears to be the *e-FtFw* model leveraging enriched category descriptions. Indeed, the corresponding job categorization is the most accurate with a relaxed error rate E_2 of almost 20%, which is satisfactory given the granularity of the chosen nomenclature (with 531 categories), so that the *e-FtFw model* has been implemented and is now daily used to categorize thousands of jobs. We see also that the simple application of the Co-Training is not an efficient way to leverage the corpus $\mathcal{J}$. A strategy could be to design another co-training process adapted to our learning-to-rank model, since the Bi-ranker model gives good results and

	E_1	E_2	E_{sim}	*AUC-ROC*
CM	0.38	0.33	0.39	0.766
Fw	0.31	0.27	0.36	0.842
FtFw with Forest	0.29	0.25	0.31	0.872
FtFw	0.28	0.24	0.34	0.862
bi-ranker	0.27	0.23	0.32	0.870
co-training	0.30	0.28	0.43	0.825
e-FtFw	**0.25**	**0.21**	**0.25**	**0.880**

Table 4.3 – Performance of the proposed approach (e-FtFw) compared with the alternatives, on a set of 1,339 job adverts and 531 job categories.

was designed to validate the assumptions of the co-training. Furthermore, the *Bi-ranker* performance confirms firstly that the FtFw model still works when learned only on $\boldsymbol{j_1}$ and only on $(\boldsymbol{j_2}, \boldsymbol{j_3}, \boldsymbol{j_4})$, and secondly that our probability estimate for the *FtFw* is precise enough to combine the two rankers like in Equation (4.18). Over all, the poor results of the *Co-training* teach us that in this standardization problem, it is more efficient to enrich the category descriptions of $\mathcal{C}$ instead of enriching the training set Γ: the category descriptions are critical and are potentially improvable, whereas the similarity function $sim_\lambda(j, c)$ does not seem to be improvable by learning it on more examples. This result confirms that the textual descriptions of a nomenclature are at the core of the standardization problem.

4.7 The Importance of a Nomenclature with External Knowledge

In this chapter, we have used the textual data associated to the nomenclature $\mathcal{N}$ (written $\mathcal{C}$ for job categories) that was predefined by domain experts. This external knowledge has been at the core of our standardization, when computing the similarity, and moreover the knowledge enrichment has shown great results. In particular, while we have failed to improve the standardization function f through co-training - our baseline of semi-supervised learning - we managed to improve the standardization precision by enriching the category descriptions of $\mathcal{N}$. Beyond standardization itself, the nomenclature's meta-data also finds several practical applications, like for user experience - when manipulating objects with textual descriptions - and the international relations that link a French job category to its American equivalent.

Unfortunately, only few nomenclatures directly usable exist, and the job categories will remain our only example of ready-to-use nomenclature for the e-recruitment domain. In light of this, we will study in the next chapter the feasibility to build a generic nomenclature $\mathcal{N}$; a special focus will be to leverage

external knowledge extracted from the Semantic Web (see Section 2.2.1) as well as from the social media. These sources will be at the core of the construction of $\mathcal{N}$, will help at being generic, and will provide crucial textual data that will be at the core of standardization.

Chapter 5

Generating Knowledge Bases Usable as Nomenclatures

The nomenclature $\mathcal{N}$ and its associated knowledge appears to be crucial for standardization, but for most of the concepts - for example the skills and the educational institutions - there is no ready-to-use nomenclature with rich meta-data. In this chapter, we study the feasibility to construct automatically a nomenclature of entities $\mathcal{N}$; in this case, each document to standardize is ideally represented by one of the entities, and the standardization function f aims to find the right one. As observed previously, the core of an efficient standardization will be the knowledge associated to the nomenclature, that we will extract from public sources. These are external knowledge bases, which present a generic purpose while the nomenclature for standardization concentrates by definition on a domain. After having detailed the standard process for this construction, we will study its instantiation in the case of the skills and the educational institutions, which each will raise different questions:

- When considering a generic knowledge base, how to select the appropriate entities for $\mathcal{N}$ in the case of a concept difficult to define, like the skills, which results from a common knowledge? For this example, the construction needs to follow the job market trend, expressed by changes in the skills themselves as well as in the terminology used by candidates and recruiters. The approach we proposed in [Malherbe and Aufaure, 2016] considers not only the semantic web through the DBpedia project, but also social media, incarnated by the Q&A forum Stack Overflow and a corpus of candidate profiles from professional social networks.

- As multiple public sources are often necessary to cover a sufficient range of entities, how to evaluate the merge of multiple knowledge bases when building the nomenclature? Standardizing the educational institutions requires in particular to deduplicate three complimentary bases that present numerous entities in common. In light of this, we propose a unified evaluation that considers a set of real-world queries for standardization, and efficiently produces metrics for the data matching and the entity linking.

While both problems require to focus on the nomenclature of entities, the approaches present two different difficulties. They have been implemented, and are used every day for standardizing thousands of candidate profiles and job adverts.

5.1 The Standard Process for Standardizing Entities

In this section, we motivate our strategy for standardizing entities, which is to construct a nomenclature of entities from public knowledge bases, and propose a unified approach for it, divided in three processes.

5.1.1 Advantages of a Rich Knowledge Base

By definition, an entity is a concept that *exists*, and can therefore theoretically be enumerated. For instance, the *Company* and the educational *Institution* in the e-recruitment documents are entities, contrary to the job categories that are higher-level concepts. For such textual field, standardizing means finding the right representant of a document into a nomenclature of entities $\mathcal{N}$. An entity having one or several explicit names, the task is in practice to detect the presence of the entities names inside a document $\boldsymbol{d}$, with possibly a disambiguation. This type of standardization presents many advantages when the nomenclature presents rich meta-data, in which case we consider it as a *knowledge base* - if this type of knowledge bases have generally information encoded through RDF data (see Section 2.2.1), in this chapter we will simply see them as a *flat set of entities* with meta-information. A first avantage of the knowledge bases is when each entity is associated to several aliases, so that standardization performs a deduplication, meaning that several different but equivalent wordings refer to the same concept. Another advantage is when the nomenclature is multilingual, meaning the entities have aliases in different languages. In this case, standardization is equivalently performed on documents of different languages, so that the final statistics can include several countries; for example, when a job advert in French talks about "soudage", we can compare it to a job advert in English talking about "welding". Besides, when the entities of the nomenclature are classified or organized as a taxonomy, it is possible to regroup the entities into meaningful groups. Moreover, when a description is associated to each element of $\mathcal{N}$, it improves the user experience, as well as the taxonomy when exploring the knowledge base.

Beyond these aspects, the richness of the nomenclature is crucial for making standardization efficient. Indeed, the knowledge base used needs to be the most comprehensive possible in terms of number of entities contained, in order to standardize the larger number of documents. Furthermore, the meta-data is associated to each entity also increases the coverage, but also the standardization precision, because the more the aliases there are, the finer the deduplication is. In light of this, for standardization purpose only, the aliases of the entities of $\mathcal{N}$ will be the most crucial data. Other meta-data can however be used by

the system, for instance during the disambiguation of the possible entities for a document $\boldsymbol{d}$.

With these considerations in mind, we dissect in the following sub-section a unified process for building a knowledge base from public sources, in order to use it as a nomenclature.

5.1.2 The Three Processes of this Type of System

With the growth of the Internet, many knowledge bases exist (see Section 2.2.1) and can serve as a basis for generating a nomenclature for a given standardization problem. Unfortunately, it is rare to have a knowledge base directly usable as a nomenclature, as was the case for the job categories. Indeed, some bases are generic and contain many concepts, whereas a nomenclature is specific to a domain by definition. On the other side, some knowledge bases are not comprehensive enough and would not constitute a sufficiently representative nomenclature when considered alone. In light of this, when standardizing entities, we generally need to construct the nomenclature based on several sources of knowledge. Designing the standardization system involves therefore the three following processes:

1. **Selecting** the relevant sources of knowledge and the respective entities: we first need to constitute a pool of entities that we extract from public knowledge bases. On one side, we have to manually inspect and choose which sources are relevant or not for our system, depending on the industrial use case, like the language of the documents to be standardized. On the other side, since these sources might be too generic and deal with irrelevant entities, we need to select - ideally automatically - the sub-part of these knowledge bases that corresponds to our current attribute.

2. **Merging** the sources to produce a deduplicated knowledge base: this is necessary when several sources have been selected. In this case, the extracted knowledge bases will very probably present an overlap of entities, so that we need to merge the duplicated entities. This process might be assimilated to a data matching (Section 2.2.2) and produces the final nomenclature $\mathcal{N}$.

3. **Linking** the unstructured documents $\boldsymbol{d}$ to the corresponding entity of $\mathcal{N}$. This step corresponds to entity linking (Section 2.2.3) and the answer might be several entities, in the case of a long free text for instance. This process is in practice done by comparing $\boldsymbol{d}$ to the documents associated to the entity; since those documents each presents explicit entities name(s), this comparison is simpler than the FtFs model, which compared documents of different nature and structure, and required some supervised learning. This step is the actual standardization, and is performed at each document $\boldsymbol{d}$ to be standardized, contrary to the *selection* and *merging* that are performed only when generating/updating the nomenclature.

These processes are illustrated in Figure 5.1, from Bob's point of view. In the following of this chapter, we will present the instantiation of those steps in

two use-cases: the *Skills* present in a candidate profile or in a job advert, and the educational *Institutions* of a profile.

5.1.3 Two Types of Entities, Two Problems

A first object of study in this chapter will be the *Skills*. This kind of concept is obviously an entity in the case of a *hard skill* (see Section 1.3), for example a mechanical tool or a software. Theoretically, some soft skills could however be inferred from a document that does not explicitly state them. For example, a job advert about management could implicitly require some relational competencies without stating them; we will however forget those implicit skills and focus on the explicit ones, that are therefore assimilated to entities. The skills are indeed a major aspect when dealing with job ads or candidate profiles, but the existing knowledge bases of skills are too far from real-world terminology used by candidates [Braun et al., 2010]. Despite their costly development, they can not be efficiently used for skills extraction, for which they constitute a top-down approach since the knowledge come from recruitment experts. Building a knowledge base of skills from public sources appears necessary, but presents a difficulty when *selecting* the correct entities: what entity can be considered as a skill? This question seems to be answered differently by recruitment experts and candidates, so that we propose a bottom-up approach for selecting the entities of $\mathcal{N}$, that finds its roots in a corpus of real-world candidate profiles.

Our second object of study will be the educational *Institutions*. Present only on the resumes, this attributes is crucial: the reputation of the institutions where the candidate graduated is one of the most regarded feature by recruiters (the second one according to [Archer and Davison, 2008]). For this example of entities, the institutions are characterized by generally long, numerous and various aliases, for example the acronyms, nicknames and historical name(s) of a university. Consequently, handling them raises a difficulty for *merging* the public sources, since we need to perform a data matching, and to evaluate accurately this process. The merging has indeed great effect on the final standardization, since it influences directly the nomenclature's quality, so that we propose a unified process for evaluating the global system which produces efficiently metrics on the merging quality.

5.2 Leveraging Candidate Skill Terminology

Listing the skills required for a job or mastered by a candidate is a crucial example of textual standardization, in which several structured values are inferred inferred per document - instead of a unique as in the others standardizations treated in this thesis. The skills extraction on both side leads indeed to many applications [Trichet and Leclère, 2003], like skills-based matching between ads and candidates, experts finding systems [Riahi et al., 2012], HR management [Fazel-Zarandi and Fox, 2012], or advanced statistics on large corpuses like the emergence of skills [Abbound et al., 2015]. This standardization is also very beneficial for the SmartSearch project, for which needs to be performed both on

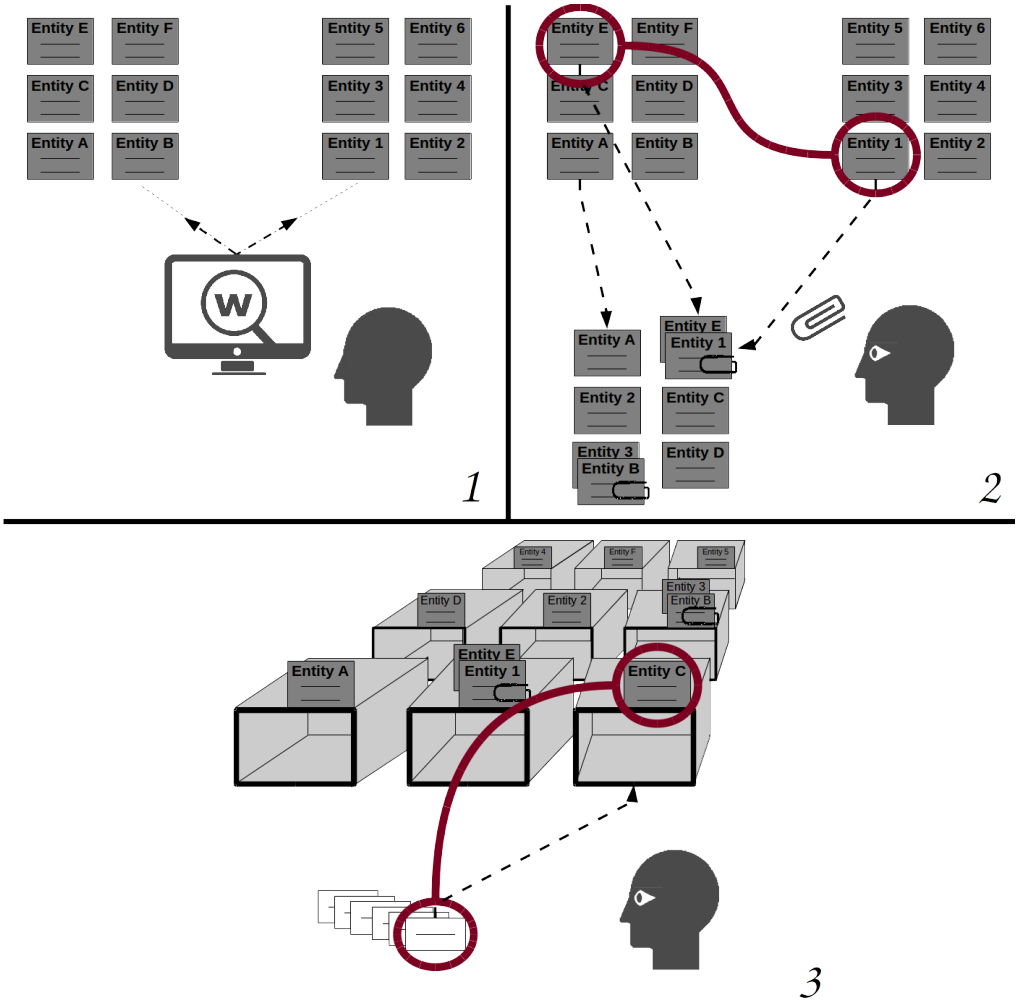


Figure 5.1 – The approach for generating nomenclatures as perceived by Bob. In the first frame, Bob downloads the appropriate entities from multiple sources (2 in this illustration), which is the selection process and generally involve filtering irrelevant entities. From each source results a set of entities, each of them being represented by a short document of meta-data, which includes a list of aliases. In the second frame, Bob performs the merging process by deduplicating the sheets of paper representing the same entity. The last frame represents the linking process, once each deduplicated entity has been associated to a box. Bob puts the sheet of paper to be standardized into the box for which an alias is in the observed sheet of paper.

resumes and job ads, in order to compare the statistics on the candidate side and on the recruiter side.

In this study, we will only consider the last field of the candidates' profiles, the **Skills**, that is a list of expressions that one could assimilate to tags. Each of those expressions is an unstructured raw text, made up possibly of several words and referring (ideally) a single entity. In the following, a candidate profile p will be simply defined as this list of expressions in *lowercase*, like in the example below:

$$p = \{\text{“adobe premiere”}, \text{“microsoft office”}, \text{“photoshop”}, \\ \text{“video editing”}, \text{“photography”}\} \quad \in \mathcal{P}_{\text{EN}}$$

where $\mathcal{P}_{\text{EN}}$ denotes the corpus of all candidate profiles in English, and $\mathcal{P}_{\text{FR}}$ the corpus of profiles in French, that have been extracted from professional social networks for the SmartSearch project. In this study, we will use the notation t for this type of expression, which might regroup several words, contrary to the rest of the thesis where a term t is a single token. Despite this aspect, the expressions t always refer to a unique concept, and could therefore be assimilated to a phrase [Arnon and Snider, 2010], which is regularly considered in the literature as a single token and is generally extracted based on statistical rules.

In this first practical example of knowledge base construction, the standardization target are candidate profiles p, as well as the textual descriptions of jobs, for which standardization is multivalued. As we will detail in the next section, the *selection* step is not straightforward since the definition of a skill is subjective and results in fact from a common knowledge. We propose therefore a new bottom-up approach (illustrated in Figure 5.2) to generate an up-to-date multilingual knowledge base of skills using social media and the terminology of skills extracted from the candidate profiles. The social media is used in two aspects during the *selection* step: firstly, professional social networks - from where candidate profiles are extracted - serve at deciding which entities to extract, and secondly, a Q&A platform is used as a public source of knowledge.

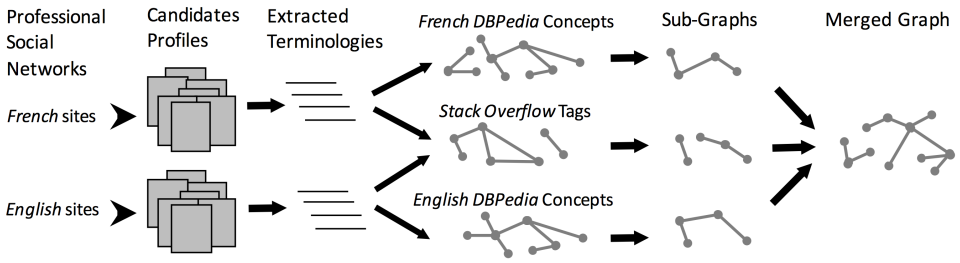


Figure 5.2 – Diagram describing the steps and data involved in building the knowledge base of skills $\mathcal{S}_{FINAL}$. All those steps but the last constitute the selection *process of the nomenclature's construction.*

5.2.1 Selecting Skills from Social Media and the Semantic Web

The public sources that appears to contain concepts related to skills are DBPedia (see Section 2.2.1), in French and English for our study, as well as the tags of Stack Overflow[1]. This Q&A website is specialized in computer science, for which it seems to be comprehensive and up-to-date; this kind of website has been found relevant for finding experts [Riahi et al., 2012] and the tags include programming languages, softwares or frameworks. However, for each of those public sources, there is no obvious way for *selecting* which entities are indeed skills. In light of this, the main aspect when building our knowledge base is the definition of a skill, which is difficult linguistically. The collaborative approach proposed in [Braun et al., 2010] points out a group effect, because the employees prefer to use frequent tags when they specify the skills of someone. The definition of a skill has therefore a social aspect, so that we investigate the use of social media.

Language	French	English
Number of Sources	12	9
Number of Profiles $\lvert\mathcal{P}_{\text{FR}}\rvert, \lvert\mathcal{P}_{\text{EN}}\rvert$	916,284	414,480
Number of Skills	5,976,950	1,548,866
Number of Distinct Skills	99,974	52,661
Number of Frequent Distinct Skills $\lvert\mathcal{T}_{\text{FR}}\rvert, \lvert\mathcal{T}_{\text{EN}}\rvert$	9,111	3,898

Table 5.1 – Characteristics of our corpuses of candidate profiles.

To benefit from the growth of professional social networks while avoiding the expensive cost of the manual tagging used in [Braun et al., 2010], we propose to consider the corpuses of candidate profiles $\mathcal{P}_{\text{FR}}$ and $\mathcal{P}_{\text{EN}}$, whose stats are in table 5.1 once the candidates having an empty **Skills** field are filtered. Our hypothesis is then to define a skill as *an expression that appears frequently in the skills field content of the candidate profiles*. In other words, only the expressions of $\mathcal{P}_{\text{FR}}$ that appear more than a given threshold ϵ are considered for the French language, which defines formally the French terminology of skills

$$\mathcal{T}_{\text{FR}} = \{\text{expression } t | \exists \text{ more than } \epsilon\% \text{ of profiles } p \in \mathcal{P}_{\text{FR}} \text{ such that } t \in p\}$$

and similarly in English for $\mathcal{T}_{\text{EN}}$. In our experiments, the threshold was fixed at $\epsilon = 0.01\%$; the motivation for this choice is to get a similar number of skills compared to the existing skills bases (see Table 5.3). In a future work, we plan to investigate a strategy to determine the optimal threshold automatically, with both qualitative and quantitative considerations. For each language, the list of expression resulting from this process forms our skills terminology, and Figure 5.3 gives an overview of $\mathcal{T}_{\text{FR}}$ and $\mathcal{T}_{\text{EN}}$ for the French and English languages. To support more languages in the final knowledge base, one simply needs to consider an additional corpus of candidates in the desired language, and apply this terminology extraction procedure.

[1] `http://stackoverflow.com/`

“Management”, “Communication”, “Gestion de projet”, “Marketing”, “Microsoft Office”, “Microsoft Excel”, “Microsoft Word”, “Informatique”, “Vente”, “Adobe Photoshop”	“Customer Service”, “Cashier”, “javascript”, “Graphic Designer”, “Project Manager”, “java”, “php”, “jquery”, “Administrative Assistant”, “Sales Person”

Figure 5.3 – Top 10 most frequent expressions, in French on the left and English on the right.

It is worth noting that each extracted terminology is a just flat set of expressions and contains some noise and duplicates. Despite being finite, this set is not usable as a nomenclature, since our skills knowledge base should have rich information (as introduced in Section 5.1.1), whereas the expressions of the terminologies come with no meta-data nor relations. We will however make use of the terminologies $\mathcal{T}_{\text{FR}}$ and $\mathcal{T}_{\text{EN}}$ for selecting the related concepts in the especially chosen knowledge graphs, DBpedia in French and English, and the tags of Stack-Overflow. As a result, we obtain the partial bases $\mathcal{S}_{\text{DBP-FR}}$, $\mathcal{S}_{\text{DBP-EN}}$ and $\mathcal{S}_{\text{S-O}}$, which are formally a set of skills s associated to a list of synonyms or *aliases*, written $aliases_{\text{FR}\to s}$ for the French aliases and $aliases_{\text{EN}\to s}$ for the English ones.

The entites selection process starts by extracting a sub-part of DBpedia that matches our skills terminology, in French and English. In this knowledge base, each entity comes with information such as a category, a description, a type, and some *aliases* corresponding to the URL redirections [Wu and Weld, 2010]. Whereas a natural approach is to select the concepts based on the “rdf:type” attribute of the DBpedia concepts - as we will use in Section 5.3.1 - the difficulty to determine what is a skill makes that there is no type dedicated to it. We

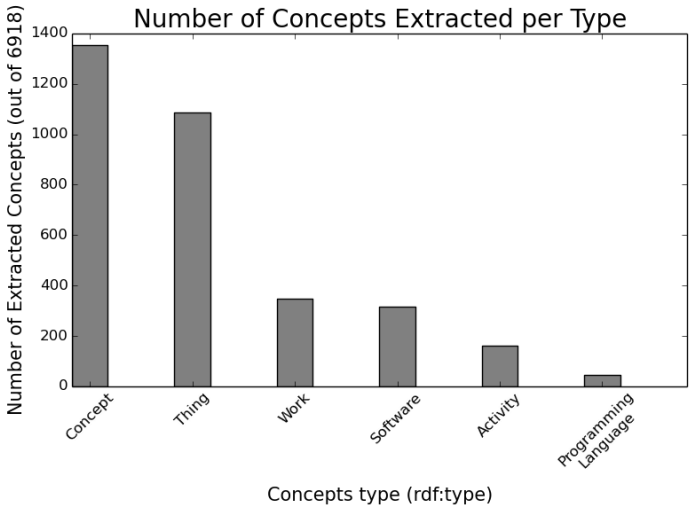


Figure 5.4 – The most frequent “rdf:type” of the skills of $\mathcal{S}_{DBP\text{-}EN}$, extracted from the English DBpedia. The extracted knowledge base is very heterogeneous and contains only few programming languages.

therefore propose to extract each entity that is named in the terminology. In other words, our approach amounts to define a DBpedia entity as a skill when it *has one of its aliases is in the skills terminology.* For each language and the corresponding DBpedia (they are distinct bases), the concepts that verify this property are extracted, which formally defines the set $\mathcal{S}_{\text{DBP-FR}}$ of skills concepts:

$$\mathcal{S}_{\text{DBP-FR}} = \{\text{concept } s \text{ of DBpedia-FR } | \exists\ t \in \mathcal{T}_{\text{FR}} \text{ such that } t \in \mathit{aliases}_{\text{FR}\to s}\}$$

and similarly in English for the set $\mathcal{S}_{\text{DBP-EN}}$. The elements of $\mathcal{S}_{\text{DBP-EN}}$ and $\mathcal{S}_{\text{DBP-FR}}$ are more than expression, since they are associated to meta-data. Table 5.4 shows the heterogeneity of the entities extracted. Any language for which DBpedia is rich enough can be considered in this process, provided we have extracted the corresponding terminology from a corpus of candidates.

Despite its richness, the set of DBpedia entities might not be comprehensive, so that we need to enrich the knowledge base of skills with extra entries. As introduced previously, we consider in our construction an additional public source, the tags of Stack Overflow. Those tags refer to an entity and are generally associated to a description, some aliases, and similarities with other tags (accessible in open access[2]). Similar to DBpedia, since only few tags really correspond to skills, we need to filter the irrelevant skills, starting by those corresponding to a natural language word, using a list of english words[3], formally written $\mathcal{W}$. Then, we consider the subgraph of the Stack Overflow tags that matches the French and the English terminologies of skills (those tags being not specific to any language), which formally defines the set:

$$\mathcal{S}_{\text{S-O}} = \{\text{tag } s \text{ of StackOverflow } | \exists\ t \in \mathcal{T}_{\text{FR}} \cup \mathcal{T}_{\text{EN}} \text{ such that } t \in \mathit{aliases}_s \text{ and } \mathit{aliases}_s \cap \mathcal{W} = \emptyset\}$$

where the aliases $\mathit{aliases}_s$ for the tag s are not specific to any language. Our study is limited to Stack Overflow, the richest forum of the Stack Exchange[4], but other forums could be relevant, like the ones dedicated to *Management* and *Graphic Design.*

The construction of $\mathcal{S}_{\text{DBP-FR}}$, $\mathcal{S}_{\text{DBP-EN}}$ and $\mathcal{S}_{\text{S-O}}$ is our *selection* process. It can be seen as a filtering of the entities, based on the candidates terminology. As introduced in Section 5.1.2, we need now to form from these three sets a unique set of entities, that will be our skills nomenclature.

5.2.2 Merging into a Final Knowledge Base of Skills

This step produces the final knowledge bases of skills $\mathcal{S}_{\text{FINAL}}$ by *merging* the subgraphs $\mathcal{S}_{\text{DBP-FR}}$, $\mathcal{S}_{\text{DBP-EN}}$ and $\mathcal{S}_{\text{S-O}}$, since they are independent and contain duplicated concepts. With this objective, the Stack-Overflow tags are merged to the DBpedia concepts when they share one alias in common, and the DBpedia concepts are merged with their equivalent concept in the other DBpedia based

[2] `https://data.stackexchange.com/`

[3] `http://www-personal.umich.edu/~jlawler/wordlist.html`

[4] `http://stackexchange.com/sites`

on the "same as" relation. One notes that other knowledge bases, such as YAGO or FreeBase, could also be easily merged, provided that there exists explicit relations linking the concepts among bases.

In this merge process, we have to carefully consider the DBpedia concepts corresponding to the *disambiguations pages*, since this kind of page precisely makes the link between several different concepts that can be confused and should not be merged. This type of concept is considered in our system as an *ambiguous skill*, except if it is linked to only one other skill of the knowledge base: it then means that the other disambiguated senses of the page are not skills. In this case, the disambiguation concept is deleted, and the corresponding skill is enriched with an additional alias, the title of the disambiguation page. The ambiguous skills represent less than 5% and are skipped for the extraction described in Section 5.2.3.

Knowledge Base	Total Number of Concepts	Skills Extracted	# aliases per Skill
French DBpedia	1,721,600	4,845	3.63
English DBpedia	5,072,562	4,691	8.53
StackOverflow	114,742	1,520	3.76
Final Knowledge Base	5,157	-	11.64

Table 5.2 – Characteristics of the different knowledge bases involved.

The knowledge base of skills $\mathcal{S}_{\text{FINAL}}$ we obtain contains rich information, extracted from the initial knowledge bases. Firstly, a significant information about each skill s of the knowledge base are its aliases (as detailed in Section 5.1.1), that is formally expressed as a set of lowercase expressions $aliases_{\text{EN}\to s}$ for English, and $aliases_{\text{FR}\to s}$ for French. As an example, the skill s for "Mobile Marketing" is associated to the following languages:

$$\begin{aligned} aliases_{\text{EN}\to s} = \{&\text{“mobile marketing”, “sms marketing”} \\ &\text{“mobile interaction service”}\} \\ aliases_{\text{FR}\to s} = \{&\text{“marketing mobile”, “m-marketing”}\} \end{aligned}$$

Secondly, each skill is associated to a textual description, that we display in SmartSearch's interface in order to improve the user experience. Considering the same example as above, the description starts with *"Mobile marketing is marketing on or with a mobile device, such as a smart phone."*.

Another relevant information are the categories of the DBpedia concepts. Those categories constitute a taxonomy, that be used for exploring the knowledge base of skills. For the SmartSearch project, we opted for a limited number of categories that we manually selected, in the idea of grouping the skills into skills-related categories. Among the few dozens of categories we considered, there are "Law", "Software" and "Marketing" (in which the previous example, "Mobile Marketing", is categorized). To extend these categories to the skills obtained through StackOverflow, we simply analyzed the first sentence of the description,

and assign a type if some specified keyword is found, such as "framework" or "programming language". This way, the skills of $\mathcal{S}_{\text{FINAL}}$ are automatically clustered into macro categories. We can therefore work out statistics grouped per category, and concentrate on specific categories.

These meta-data will be used in the SmartSearch project, as illustrated later in chapter 7. In this chapter, we will only use the aliases, for standardization purpose as detailed in the following.

5.2.3 Linking Candidates and Jobs to the Corresponding Skills

This section explains the *linking* process resulting from the knowledge base $\mathcal{S}_{\text{FINAL}}$, which is straightforward and only uses the aliases of the skills. It serves on the candidate side as well as the job side, for which it returns multiple structured values.

A first natural use of the knowledge base is to associate a candidate profile p to its corresponding skills. To do so, each expression $t \in p$ is linked to the skill s of the knowledge base that contains the expression t in its aliases $aliases_{\text{FR}\to s}$ (if case of a profile coming from a French website, for instance), if this skill exists. Formally, this defines a standardization function

$$normalize_{\text{FR}}(t) = \begin{cases} s & \text{If } \exists\ s \in \mathcal{S}_{\text{FINAL}} \text{such that } t \in aliases_{\text{FR}\to s} \\ \text{NIL} & \text{Otherwise} \end{cases}$$

This standardization is a variant of the wikification (see Section 2.2.3), and provides various advantages: each standardized expression t is linked to some meta-data that was detailed in Section 5.2.2; the expressions representing the same skill are deduplicated, even for expressions in different languages; last, this standardization filters the expressions that are absent from the knowledge base $\mathcal{S}_{\text{FINAL}}$, meaning they are too infrequent in $\mathcal{P}_{\text{EN}}$ and $\mathcal{P}_{\text{EN}}$ or absent in DBpedia and StackOverflow.

The second and main application of this newly created nomenclature is to extract skills of a job advert, which is a raw unstructured text. We simply write this text $\boldsymbol{d}$, but contrary to the other chapters, we do not extract only the single tokens but all the 1 to 4 grams of words from $\boldsymbol{d}$. This multiset of n-grams of words is written $ngrams(\boldsymbol{d})$, and corresponds to a process similar to [Abbound et al., 2015] where they propose to look at the frequent and emerging n-grams in a data-set of job ads to detect the trendy skills. Contrary to their approach, our process provides structured data, by extracting the skills of $\mathcal{S}_{\text{FINAL}}$ having one of its aliases in the set $ngrams(\boldsymbol{d})$. Formally, this defines an extraction function

$$extract_{\text{FR}}(\boldsymbol{d}) = \left\{ s \in \mathcal{S}_{\text{FINAL}} \,\middle|\, ngrams(\boldsymbol{d}) \cap aliases_{\text{FR}\to s} \neq \emptyset \right\}$$

An example of extraction is given in Figure 5.5, using the knowledge base $\mathcal{S}_{\text{FINAL}}$ on a job advert found on *Indeed.com*. This example shows that extracting the skills gives a summarization of the job advert, and can be assimilated to automatic tagging. Furthermore, most of the extraction covers the second paragraph, which talks about the tasks of the job, contrary to the first one which introduces

the context of the post. In the real-world usage of standardization (detailed later in Chapter 7), this job advert would be for instance counted in the statistics about the skill "Adobe Indesign".

Job Title and Description:

Russian speaking Web Designer, London
A great new opportunity has just become available for an experienced Russian Speaking Senior Web Designer to join international company and become part of their London team. If you are interested in this position yourself or know someone who might be, please contact us.
Required skills: Strong knowledge of the Adobe Suite - Photoshop, Indesign and Illustrator; Strong visual eye and attention to detail; Understanding of UX principles and User Centered Design ; Ability to take a PSD and turn it into a functioning prototype; Visual design of user interfaces for websites.

Extracted Skills $extract(\boldsymbol{d})$**:**
designer / design / website / speech / web design / adobe photoshop / russian language / user (computing) / prototype / adobe indesign / illustrator / availability / user-experience / international / adobe / unix / user interface / information technology

Figure 5.5 – Example of skills extraction on a real-world job advert.

5.2.4 Experimental Comparison with Other Knowledge Bases

To justify the use of our newly generated nomenclature of skills and evaluate quantitatively the standardization we obtain, we computed several metrics by using $\mathcal{S}_{\text{FINAL}}$ as well as other knowledge bases of skills. There already exists knowledge bases, private or open, that are directly usable as a nomenclature for our system. Using one of these existing knowledge bases of skills extraction is a top-down approach in the sense that the skills are pre-defined by professionals, and do not necessarily correspond to the terminology used by candidates and recruiters. In our experiments, we considered the following alternative knowledge bases:

- *O*Net* [Hilton et al., 2010]: as introduced in Section 2.2.1, the O*Net knowledge base is used by the U.S. public recruitment services. The core concepts are the job categories, that are associated to generic skills as well as "tools and technology". The skills nomenclature is very coarse with only 35 skills. We instead extracted the list of the "tools and technology", constituting a large set of skills with only few meta-data and in English only.

- *SkillsPlex*[5]: this private knowledge base has been used at SAP for years, mainly for human resources management. It is only in English and contains detailed descriptions for thousands of skills.

[5] `www.peoplesciences.com/psilibrary.htm`

- *ESCO* [De Smedt et al., 2015]: the European Skill, Competences and Occupations taxonomy is an thesaurus freely available in 10 languages. From this knowledge base introduced in Section 2.2.1, we extracted only the information related to skills, namely several aliases in multiple languages but no textual description.
- $\mathcal{S}_{FINAL}$: This knowledge base is obtained after the process described in the sections 5.2.2 and 5.2.1.
- $\mathcal{S}_{DBP}$: This knowledge base corresponds to the process described in the sections 5.2.2 and 5.2.1, without considering the Stack-Overflow data, $\mathcal{S}_{\text{S-O}}$. In other words, this nomenclature is the merge between $\mathcal{S}_{\text{DBP-FR}}$ and $\mathcal{S}_{\text{DBP-EN}}$, with only DBpedia concepts. Testing this knowledge base of skills serves at justifying the use of Stack Overflow tags in our process.

Knowledge Base	O*Net	SkillsPlex	ESCO	$\mathcal{S}_{\text{DBP}}$	$\mathcal{S}_{\text{FINAL}}$
# skills	27,025	14,195	5,096	4,352	5,157
# aliases/skill	1	1	4.5	11.4	11.6
Skills with a description (%)	0	100	0	78	81
Skills with a category (%)	100	100	100	83	71
Multilingual skills (%)	0	0	95	78	83

Table 5.3 – Quantitative comparison of the knowledge bases of skills

Some statistics about these knowledge bases are displayed in table 5.3. In order to compare the standardization itself, we also computed several quantitative metrics to evaluate $normalize(t)$ and $extract(t)$ using the different knowledge bases of skills. We first computed the coverage for the normalization $coverage_{\mathcal{P}} \in [0, 1]$ on the test sets of candidates $\mathcal{P}_{\text{EN}}$ and $\mathcal{P}_{\text{FR}}$. This quantity is the proportion of expressions $t \in p \in \mathcal{P}_{\text{EN}}$ that are standardized by $normalize(t)$, and is formally given by

$$coverage_{\mathcal{P}} = \frac{\sum_{p \in \mathcal{P}_{\text{EN}}} \sum_{t \in p} \mathbb{1}(normalize(t) \neq \text{NIL})}{\sum_{p \in \mathcal{P}_{\text{EN}}} |p|}$$

where $\mathbb{1}(normalize(t) \neq \text{NIL})$ denotes the identity function that equals 1 if the expression e is normalized or 0 otherwise. One notes that this coverage is computed on the sets used in Section 5.2.1, but is counted with expressions frequency and before any filtering. Consequently, we expect to obtain a high coverage on the profiles using $\mathcal{S}_{\text{FINAL}}$, that was the initial aim of our construction.

A more interesting and unpredictable behaviour is the standardization of job descriptions. Indeed, they are written by recruiters, contrary to the candidate profiles $\mathcal{P}_{\text{FR}}$ and $\mathcal{P}_{\text{FR}}$ used to construct $\mathcal{S}_{\text{FINAL}}$. To evaluate this second standardization, we computed the absolute coverage $coverage_{extract}$ for the extraction as the average number of skills extracted from a job advert $\boldsymbol{d}$, formally written as

$$coverage_{\mathcal{J}} = \frac{\sum_{\boldsymbol{d} \in \mathcal{J}} |extract(\boldsymbol{d})|}{|\mathcal{J}|}$$

where $\mathcal{J}$ is the sample of job adverts. Despite the notation, this corpus of jobs differs from the one used in Chapter 4, because each job is described by only one document $\boldsymbol{d}$, obtained after concatenation of the job *Title* and *Description*. Unlike $coverage_{\mathcal{P}}$, the value for $coverage_{\mathcal{J}}$ is absolute, because we can not divide it by the number of skills to be ideally extracted, this number being very subjective.

Lastly, the extraction precision $precision_{\mathcal{J}} \in [0, 1]$ was estimated on a sample J of 100 job adverts. In order to represent a wide range of jobs, each advert of $\mathcal{J}$ was taken from a different job category, using the automatic categorization described in Chapter 4. For each job advert, a team of Multiposting experts have manually labeled the extracted skills as being relevant or not. The set of relevant skills for a job advert $\boldsymbol{d}$ is written $valid(\boldsymbol{d})$, and the precision is formally given by

$$precision_{\mathcal{J}} = \frac{\sum_{\boldsymbol{d} \in \mathcal{J}} |valid(\boldsymbol{d}) \cap extract(\boldsymbol{d})|}{\sum_{\boldsymbol{d} \in \mathcal{J}} |extract(\boldsymbol{d})|}$$

which is the proportion of valid skills that have been extracted on the sample of job adverts $\mathcal{J}$. For this last metric, the set $\mathcal{J}$ of job adverts is smaller, due to the cost of the manual validation.

Metric	Tested Knowledge Base	O*Net	SkillsPlex	ESCO	$\mathcal{S}_{\text{DBP}}$	$\mathcal{S}_{\text{FINAL}}$
$coverage_{\mathcal{P}}$ (%)	5,976,950 expressions from French profiles	-	-	13.9	75.9	76.9
$coverage_{\mathcal{P}}$ (%)	1,548,866 expressions from English profiles	2.5	20.7	16.1	65.7	73.8
$coverage_{\mathcal{J}}$	100,000 French adverts	-	-	2.9	13.0	15.1
$coverage_{\mathcal{J}}$	100,000 English adverts	1.2	11.8	11.6	31.6	35.2
$precision_{\mathcal{J}}$ (%)	Skills Extracted	-	-	73.9	-	81.0

*Table 5.4 – Results of the experiments on the different data-sets, using the 5 skills knowledge bases. The knowledge bases SkillsPlex and O*Net are in English, so that they were only tested on the English data-sets.*

The results in table 5.4 show that for the two languages, the *O*Net*, *SkillsPlex* and *ESCO* knowledge bases poorly cover the candidates vocabulary, unlike $\mathcal{S}_{\text{FINAL}}$ which get a high $coverage_{\mathcal{P}}$. This was expected since the $\mathcal{S}_{\text{FINAL}}$ has been constructed based on these candidates, but the comparison shows the poor results of the top-down approach using pre-defined bases. The coverage on the job descriptions $coverage_{\mathcal{J}}$ is an unbiased metrics for which the $\mathcal{S}_{\text{FINAL}}$ gets also the highest coverage, meaning it captures well the vocabulary used by recruiters - even if it is built from the candidates terminology. We note in both cases that $\mathcal{S}_{\text{DBP}}$ gets lower coverage values, justifying therefore the use of Stack Overflow in our entities selection process.

In terms of precision, the value for $precision_{\mathcal{J}}$ appears to be satisfactory, with more than 80% using our final system. The quality of the extraction is directly linked to the quality of the aliases of a skill, for instance $aliases_{\text{FR}\to s}$ in French. Indeed, let us consider an alias $t \in aliases_{\text{FR}\to s}$ that is poorly related

to the skill s, meaning that a job described by $\boldsymbol{d}$ containing the alias t does not actually relate to the skill s. However, since $t \in ngrams(\boldsymbol{d})$, skill s would be extracted from the job, which is an undesirable extraction. Based on this fact, we deduce that the aliases in the knowledge base *ESCO* are of lower quality than in the nomenclature $\mathcal{S}_{\text{FINAL}}$. In practice, we even observed that different skills from *ESCO* can share the same alias, whereas in $\mathcal{S}_{\text{FINAL}}$ the aliases correspond to an URL and are therefore singular.

The experiments confirm that the knowledge base created through this section is exploitable as a nomenclature for standardization purpose. The *selection* step we propose, based on social media, appears to consider relevant entities for the considered use case. The construction process can be repeated to benefit from an updated corpus of candidate profiles, as well as the evolution of the public knowledge bases that are both maintained by the crowd of internet users. We can therefore update the knowledge base of skills and follow the evolution in the job market, or support a new language for standardization. In this knowledge base construction, the *merging* process was straightforward, since we used the interconnexion among DBPedia projects, as well as the full names of a skill, that are simple expressions with one or two words in general. For other kind of entities, like the educational **Institutions**, the *merging* of the sources might be more subtle than for the skills. In this case, the nomenclature construction highly depends on the data matching process, which directly influences the knowledge base quality as we will see in the following section.

5.3 Evaluating the Merge of Knowledge Bases of Institutions

In the following, we study standardization of the **Institution** field content of a resume, for which three examples are displayed in Figure 5.6. An educational institution is easy to define, contrary to the skills, so that few satisfactory knowledge bases of schools already exist and provide many synonyms for each entity. However, while the *selection* process is eased, the *merging* and *linking* processes are complicated, due to the high variance in an institution names, for example "UCB", "University of California, Berkeley" or "Berkeley, Cal". The former process will be tackled by data matching and the latter by entity linking, and while the literature always addresses these subjects separately (see Sections 2.2.2 and 2.2.3) they both influence the performance of the final standardization. In particular, a poor data matching strategy could produce entities of low quality, by associating multiple non-related institutions or keeping many duplicates in the final knowledge base. This section studies the influence of the *merging* strategy on standardization, and shows how evaluating the final standardization gives a feedback both on the entity linking and the data matching quality. The unified evaluation process we propose only requires a limited number of manual validations, and provides a labeled data-set for data matching very efficiently, on which metrics for the data matching can be computed independently from the entity linking metrics, and are highly related to the real-world application of the

Paradise City	Green University
	Master of Business Administration
Austin	Gray College
	B.Eng. in Chemical Industry
Springfield	Public High School of Springfield
	HS with Honors

Figure 5.6 – Example of educational institution in a real-world resume.

system, since they focus on institutions occurring the most.

Before detailing this novel evaluation process, this section describes first the *selection*, the *merging* and the *linking* processes involved for standardizing the Institutions.

5.3.1 Selecting Complementary Knowledge Bases of Institutions

To build a knowledge base of educational institutions, we firstly considered the entities classified with the type *"Educational Institution"* in the French and English DBpedia. This type indeed covers any kind of educational institution, so that we can use it to filter the relevant entities for the current standardization problem. For each of those selected entity, we extract the institution's label, the official website URL, the locations (expressed in text format), the number of students and diverse aliases from the Wikipedia's redirections, disambiguates pages and international links, each feature being reliable information because it is structured on the original Wikipedia. While we aim to be comprehensive about French education, we noted that most of small institutions do not have a Wikipedia page, so that we also considered the source Letudiant[6], a national website about French institutions. We extracted from each public page of Letudiant the label of the institution, the official website URL, the location (expressed again in text format) and two aliases from the postal address and the webpage's URL. The necessity to deal with multiple sources and complementarity of the presented knowledge bases will be confirmed quantitatively in Section 5.3.5 (tables 5.5 and 5.9): the DBpedia knowledge bases are less comprehensive but international, and provide richer information than to Letudiant. This former source is however more comprehensive for French national institutions, but is restricted to them. Following this *selection* of entities, we formally define an educational institution e as

$$e = \{website, aliases, \boldsymbol{locations}, students\} \tag{5.1}$$

where *website* is the URL for the institution official website, possibly null if it has not be specified; the set *aliases* regroups the documents noted ***alias*** referring to the institution e in any of the considered languages, namely French and English in this study; ***locations*** is a document regrouping the institution's locations

[6] `www.letudiant.fr`

names; *students* $\in \mathbb{N}$ is the number of students of the institution. The entity e is also given a unique name, taken arbitrarily among the names of each source, that will be used later in the user interface. We write $\mathcal{E}_{\text{DBP-FR}}$, $\mathcal{E}_{\text{DBP-EN}}$, $\mathcal{E}_{\text{LETUD}}$ the sets of educational institutions respectively extracted from the French DBpedia, from the English DBpedia, and from Letudiant.

In this use case, the *selection* of entities is simpler than for the skills, mainly because an educational institution is well defined and thus well specified in the sources. On the other side, the institutions' aliases are longer and more numerous, which will complicates the following *merging* process.

5.3.2 Data Matching for Merging the Sources

As illustrated in Figure 5.7, the data matching process aims to *merge* duplicates that represent the same institution, by testing all the pairs of entities e, e' from distinct sources, where each e represents an institution as defined above. Although we do not test pairs from the same source, the matching we propose is not exactly unique (not one-to-one) since we assume that several entities of one source can be linked to the same entity of another source. In our deterministic matching strategy, we propose to merge two entities e, e' if one of the following rules is satisfied:

1. e and e' have the same website

2. If one website is not specified, and e, e' share one alias in common

3. If one website is not specified, e, e' share one location in common, and an alias of the one is contained in an alias of the other

where for the last condition, instead of using the textual document ***locations*** as defined in Equation (5.1), we preferred to use the locations in format latitude, longitude, that are provided indirectly by the initial sources of knowledge through the geonames and the postcodes. This data matching procedure results in the final knowledge base of institutions, written $\mathcal{E}_{\text{FINAL}}$. To merge e and e', we consider the union of all aliases, that is to say $aliases \cup aliases'$, and similarly for the locations. For students, we keep the maximum number, $max(student, student')$. The rules presented here are purely deterministic, the strategy itself being not the focus of this section, but the evaluation process behind it. Indeed, based on the evaluation described later in Section 5.3.4, we have been able to tune manually the data-matching, and propose the rules above. Similarly, the labeled data-set generated in the evaluation is totally usable as a training set in the case where the data matching is performed through machine learning.

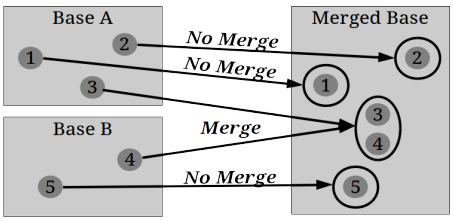


Figure 5.7 – Diagram describing the data matching process. The entities of the merged knowledge base, written E, are represented into circles because they come from the union of entities e of A and B.

We propose to step back from the previous representation of institutions, in order emphasize the conceptual difference between the entities of the initial sources and the ones of the final knowledge base, resulting from the data matching. Indeed, an entity of $\mathcal{E}_{\text{DBP-FR}}$, $\mathcal{E}_{\text{DBR-EN}}$ or $\mathcal{E}_{\text{LETUD}}$ is considered to be *primary*, and is written e, whereas an entity of the merged base $\mathcal{E}_{\text{FINAL}}$ is written E, since it is formally a set of primary entities $e \in E$. For instance, writing $E = \{e, e'\}$ means that E comes from the merge of the primary entities e and e', and the attributes of E are given by aggregating the respective attributes of e, e'. In many cases, $E = \{e\}$ is made up of a single primary entity, because the primary entity e has not been merged during the data matching process. Following this consideration, in Figure 5.8 and 5.9, a primary entity is represented by a number, while an entity E is represented by a circle grouping numbers. We emphasize the fact that *only the primary entities e are constant* because the initial sources ($\mathcal{E}_{\text{DBP-FR}}$ $\mathcal{E}_{\text{DBP-EN}}$ and $\mathcal{E}_{\text{LETUD}}$) are fixed, whereas the final entities $E \in \mathcal{E}_{\text{FINAL}}$ depend on the data matching process.

To illustrate the entities after the data matching process, here is an example of institution E:

Institution:	**Pierre and Marie Curie University**
Locations:	Paris, France
Students:	32,000
Website:	www.upmc.fr
Aliases:	Paris 6, UPMC, Paris Jussieu,
	University of Paris VI, Jussieu University

5.3.3 Entity Linking for Normalizing Resumes

In this section, we consider the knowledge base $\mathcal{E}_{\text{FINAL}}$ as fixed. With the objective of standardizing the institutions' names of a numeric resume, we successively detail here the 3 steps of the entity linking process: the extraction, the search and the disambiguation (Section 2.2.3).

The objective of our system being to standardize the institutions that are cited in a resume, we firstly extract a textual query $\boldsymbol{q}$ from a resume. We consider for $\boldsymbol{q}$ the bag of words given by aggregating the **Institution** and the **Location** associated to a degree, which is motivated by the fast that many institutions

names in $\mathcal{E}_{\text{FINAL}}$ contain their location. For instance, the query for standardizing the first line of the resume of Figure 5.6 would be:

$$\boldsymbol{q} = \{\!\{Green, University, Paradise, City\}\!\}$$

Apart from the lowercase, the only pre-processing of the query is to translate the acronyms. To do so, a list of acronyms is automatically found from the knowledge bases of institutions, when detecting for instance that "MIT" and "Massachussets Institute of Technology" are in the same aliases of a primary entity e. We only keep the acronyms that have a unique possible translation.

We then need to search for the institutions E that are relevant candidates to represent $\boldsymbol{q}$. To assess the relevance of E, we focus on the *aliases* and $\boldsymbol{locations}$, which are directly comparable with the query $\boldsymbol{q}$. We consider that an institution is relevant if and only if 70% of terms of the query $\boldsymbol{q}$ can be found in the set of words $aliases \cup locations$. Following this rule, we can retrieve the related universities E from the knowledge base, to form the set $Rel(\boldsymbol{q})$ of relevant institutions for the query $\boldsymbol{q}$. If $Rel(\boldsymbol{q}) = \emptyset$, meaning there is no relevant institution with respect to the selection rule, the answer of the system is NIL. The value of the threshold has been chosen experimentally, to give a low rate of NIL answers (below 25%) and a high rate of queries above threshold that are properly linked. This choice is based on the experimental evaluations.

Before giving the final answer of the system, we need to disambiguate the set $Rel(\boldsymbol{q})$ of relevant institutions by comparing each of them directly with the query $\boldsymbol{q}$. Using the cosine measure among documents (Equation (2.4) of Section 2.1), we compute for each institution $E \in Rel(\boldsymbol{q})$ its score:

$$\begin{aligned} score(\boldsymbol{q}, E) = 5 \times \max_{\substack{\boldsymbol{alias} \\ \in aliases}} cos(\boldsymbol{q}, \boldsymbol{alias}) + cos\Big(\boldsymbol{q}, \bigcup_{\substack{\boldsymbol{alias} \\ \in aliases}} \boldsymbol{alias}\Big) \\ + 3 \times cos(\boldsymbol{q}, \boldsymbol{locations}) + log(students) \end{aligned} \tag{5.2}$$

We then link the query $\boldsymbol{q}$ to the institution E having the highest value for $score(\boldsymbol{q}, E) \in \mathbb{R}$. This answer for the entity linking process is written $\hat{EL}(\boldsymbol{q})$, which is possibly NIL response, and is formally given by

$$\hat{EL}(\boldsymbol{q}) = \begin{cases} \text{NIL} & \text{if } Rel(\boldsymbol{q}) = \emptyset \\ \underset{E \in Rel(\boldsymbol{q})}{argmax}\ score(\boldsymbol{q}, E) & \text{otherwise} \end{cases}$$

Defining $score(\boldsymbol{q}, E)$ has been done after experimental evaluations, as well as the coefficients in the formula. Such choices are critical for the entity linking process, we need therefore to evaluate properly our system. Besides that, we see that the entity linking process directly depends on the knowledge base $\mathcal{E}_{\text{FINAL}}$ which results from the data matching. In order to justify our arbitrary choices of parameters in the merging and linking processes, we propose in the next section an unified evaluation process for this type of two-part system.

5.3.4 Evaluation Process for the Two-Part System

In this part, we step back from the industrial use case and propose an evaluation for a system combining a data matching with an entity linking.

Separated Manual Validation and its Limits

The described standardization system relies on a data matching and an entity linking, which both need to be designed and tuned in order to achieve better precision. Indeed, standardizing correctly a high proportion of queries implies first that the entity linking properly links each query $\boldsymbol{q}$ to the best entity E of the knowledge base $\mathcal{E}_{\text{FINAL}}$, but implies also a good data matching. Indeed, the final entities need to make sense and not be a merge of several distinct concepts; moreover, we want each concept to be represented by only one entity of the base, in order to have a unique valid standardization.

The evaluation of the *merging* and *linking* processes is thus critical when building this type of system, and it requires building a labeled data-set on which the system is tested and metrics are computed. Generating these labeled examples is necessary in the case where rules are set experimentally (like we did for the use-case described in previous sections) or when one of the processes is learned and involve machine learning. For evaluating the processes separately, we need a labeled data-set Γ_{EL} for the entity linking on one side, and labeled examples M, U for the data matching on the other side (see Sections 2.2.2 and 2.2.3 for more details). The first one is expressed as a set of entries $(\boldsymbol{q}, E, y_{\boldsymbol{q},E}) \in \Gamma_{EL}$ where $\boldsymbol{q}$ is a query, E a entity of the knowledge base, and $y_{\boldsymbol{q},E}$ equals $+1$ if E represents indeed the concept of query $\boldsymbol{q}$, -1 otherwise. We compute from this labeled data-set the accuracy, recall and f-measure for the entity linking:

$$acc_{EL} = \frac{\#\{\boldsymbol{q} \in \mathcal{Q} |\ y_{\boldsymbol{q},\hat{EL}(\boldsymbol{q})} = 1\}}{\#\{\boldsymbol{q} \in \mathcal{Q} |\ \hat{EL}(\boldsymbol{q}) \neq NIL\}} \qquad rec_{EL} = \frac{\#\{\boldsymbol{q} \in \mathcal{Q} |\ y_{q,\hat{EL}(\boldsymbol{q})} = 1\}}{\#\mathcal{Q}}$$

$$f - measure_{EL} = 2 * \frac{acc_{EL} * rec_{EL}}{acc_{EL} + rec_{EL}} \tag{5.3}$$

All these metrics for the entity linking are for the non-NIL answers, because verifying that a query $\boldsymbol{q}$ is not represented in the knowledge base is a costly task compared to the simple "correct" or "incorrect" choice, especially when $\mathcal{E}_{\text{FINAL}}$ is not fixed.

For the data matching, the evaluation requires a labeled data-set expressed usually as a set of pairs M to merge and a set of pairs U not to merge [Fellegi and Sunter, 1969]. We count firstly from M the number of entries $e, e' \in M$ that are correctly merged by the process, named the number TP of true positives, and secondly the entries $e, e' \in M$ that are not, named the number FN of false negatives. Similarly we count from U the number TN of true negatives and the number FP of false positive. From these quantities are computed the precision, recall and f-measure for the data matching:

$$prec_{DM} = \frac{TP}{TP + FP} \qquad rec_{DM} = \frac{TP}{TP + FN}$$

$$acc_{DM} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.4}$$

A straightforward evaluation would be to evaluate the processes independently by building distinct data-sets Γ_{EL}, M and U. This would be however time-consuming, and moreover we note that the concepts involved in each case are closely related: the primary entities e of M and U on one side, and the merged entities E for Γ_{EL} on the other side. Besides that, evaluating the data matching can be very inefficient because only few merges occur (see table 5.5 of Section 5.3.5), meaning that sampling randomly the pairs for the manual review would be extremely time-consuming. We propose thus to leverage the implicit feedback provided when validating the entity linking process, which leads to an unified and efficient evaluation process for such standardization system.

Manual Validation Process

As we focus on the real-world application of the two-part system, all the evaluation will be guided by a set of queries $\mathcal{Q}$ for standardization. For each query $\boldsymbol{q} \in \mathcal{Q}$, we will build the validation sets $T_{E\to\boldsymbol{q}}$, $F_{E\to\boldsymbol{q}}$, $T_{e\to\boldsymbol{q}}$ and $F_{e\to\boldsymbol{q}}$, as illustrated in Figure 5.8. The two first sets are constituted of final entities which have been validated as being good representant for $\boldsymbol{q}$ when $E \in T_{E\to\boldsymbol{q}}$, or bad representant when $E \in F_{E\to\boldsymbol{q}}$. The two last sets are for the data matching evaluation, and are made up of primary entities which have been inferred to be good representant when $\boldsymbol{q}$ for $e \in T_{e\to\boldsymbol{q}}$ or bad representant when $e \in F_{e\to\boldsymbol{q}}$. Quality metrics are directly computed from these sets and the process to build them is detailed in the following.

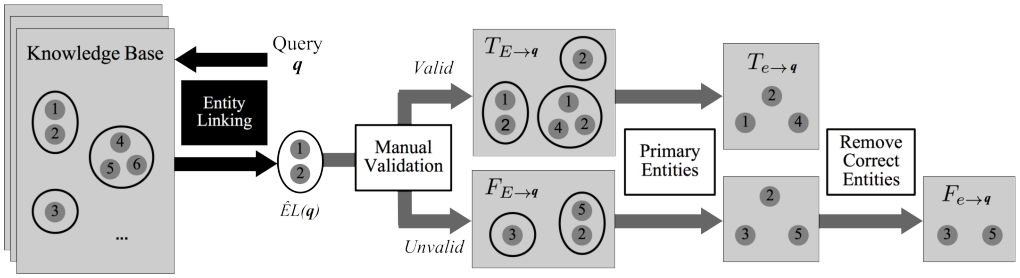


Figure 5.8 – Diagram of evaluation process with the building of sets $T_{E\to\boldsymbol{q}}$, $F_{E\to\boldsymbol{q}}$, $T_{e\to\boldsymbol{q}}$ and $F_{e\to\boldsymbol{q}}$. The number are named by numbers; circles grouping numbers represent merged entities of the knowledge base. The evaluation is performed on different knowledge bases, meaning that the entities E in $T_{E\to\boldsymbol{q}}$, $F_{E\to\boldsymbol{q}}$ result from different variants of the merging process.

Given a generated knowledge base $\mathcal{E}_{\text{FINAL}}$ and an entity linking procedure $\hat{EL}(\boldsymbol{q})$, for each query $\boldsymbol{q} \in \mathcal{Q}$ we ask for a manual validation of the suggested answers $E = \hat{EL}(\boldsymbol{q})$. This validation is done for each variant of the system being tested, which includes variants of the data matching process. For all $\boldsymbol{q}$, we build the sets of validated entities $T_{E\to\boldsymbol{q}}$ and invalidated entities $F_{E\to\boldsymbol{q}}$ in an incremental way following the algorithm 1, both being sets of merged entities

E. One notes that the entities E involved here are not the same when the data matching is modified.

Algorithm 1: Manual Validation Routine: $T_{E\to\boldsymbol{q}}$ and $F_{E\to\boldsymbol{q}}$ updates

```
input: Test-set Q, sets T_{E→q}, F_{E→q}
Output: Updated sets T_{E→q}, F_{E→q}
foreach query q in Q do
    E = ÊL(q)
    if E ≠ NIL and E not in T_{E→q} ∪ F_{E→q} then
        ask for a manual validation
        if E is a correct representant for q then
            add entity E to the set T_{E→q}
        else
            add entity E to the set F_{E→q}
        end
    end
end
```

At this point, one can directly compute the quality metrics for the entity linking based on the sets $T_{E\to\boldsymbol{q}}$ and $F_{E\to\boldsymbol{q}}$. For a given variant of the two-part system, the labeled data-set Γ_{EL} is given by simply re-expressing the previous validation as a set of entries $(\boldsymbol{q}, E, y_{\boldsymbol{q},E})$ where the query $\boldsymbol{q}$ is taken from $\mathcal{Q}$, $E = \hat{EL}(\boldsymbol{q})$ is the corresponding answer of the system, and $y_{\boldsymbol{q},E} \in \{+1, -1\}$ is the manual validation we infer in accordance with $E \in T_{E\to\boldsymbol{q}}$ or $E \in F_{E\to\boldsymbol{q}}$. The number of entry in this labeled data-set Γ_{EL} should be equal to the size of $\mathcal{Q}$ provided that the manual validation process has been fully completed for the current variant of the system. The metrics for the entity linking quality (Equation (5.4)) are directly computable from this data-set.

Implicit feedback for the merging

We propose now to leverage the previously built sets $T_{E\to\boldsymbol{q}}$ and $F_{E\to\boldsymbol{q}}$ to infer a labeled data-set for the data matching, without requiring any additional manual validation. Our approach relies on the implicit feedback about the primary entities that the manual validation provides. The idea is that when an expert validates a merged entity $E = \{e, e'\}$, this implies implicitly that e and e' represent the same concept. We consider thus the set of good primary entities for $\boldsymbol{q}$, denoted $T_{e\to\boldsymbol{q}}$, as the union of all primary entities that are in $T_{E\to\boldsymbol{q}}$. Formally it is defined as:

$$T_{e\to\boldsymbol{q}} = \bigcup_{E\in T_{E\to\boldsymbol{q}}} E = \left\{ e \in E \;\middle|\; E \in T_{E\to\boldsymbol{q}} \right\}$$

We will similarly build the set of primary entities that does not represent the query $\boldsymbol{q}$. To do so, we consider the primary entities contained in $F_{E\to\boldsymbol{q}}$; however, this set can still contain valid primary entities for the query $\boldsymbol{q}$. Indeed, let us consider $E = \{e, e'\}$ where e is valid for $\boldsymbol{q}$ and e' is invalid. Such E will be refused during the manual validation, so that the valid primary entity e will be

added to the set $F_{E\to\boldsymbol{q}}$. In other terms, the primary entities in a refused E are not necessarily all invalid for the query $\boldsymbol{q}$, but at least one. We thus remove the validated primary entities $T_{e\to\boldsymbol{q}}$ when building $F_{e\to\boldsymbol{q}}$:

$$F_{e\to\boldsymbol{q}} = \bigcup_{E\in F_{E\to\boldsymbol{q}}} E \setminus T_{e\to\boldsymbol{q}} = \left\{ e \in E \;\middle|\; E \in F_{E\to\boldsymbol{q}} \text{ and } e \notin T_{e\to\boldsymbol{q}} \right\}$$

A discussion is needed to clarify the assumption behind this definition. When removing the set of validated entities $T_{e\to\boldsymbol{q}}$, we have no assurance that it removes all valid primary entities for query $\boldsymbol{q}$, because $T_{e\to\boldsymbol{q}}$ might not be comprehensive. In other terms, when building the set of invalidated primary entities $F_{e\to\boldsymbol{q}}$, we assume that *without any positive information about a primary entity e of $F_{E\to\boldsymbol{q}}$, e is an invalid representant for $\boldsymbol{q}$*. Formally, the positive information is expressed in the set $T_{E\to\boldsymbol{q}}$, and not having any positive information about e is equivalent to $e \notin T_{E\to\boldsymbol{q}}$.

An undesired consequence of this assumption is that $F_{e\to\boldsymbol{q}}$ may contain primary entities e valid for $\boldsymbol{q}$. This occurs if an entity E containing a valid primary entity e is invalidated, and e is never validated when evaluating all the variants for the system. Nevertheless, this occurs very rarely in practice, and the consequences are negligible, because the pairs we will consider in the following are only taken in $T_{e\to\boldsymbol{q}} \times F_{e\to\boldsymbol{q}}$. For instance, when $T_{E\to\boldsymbol{q}} = \emptyset$, we will not infer any pairs for the labeled data-sets M, U of the data matching. Besides that, we can assume that only few primary entities are valid for the query $\boldsymbol{q}$, and consequently the set of validated primary entities $T_{e\to\boldsymbol{q}}$ is rapidly comprehensive.

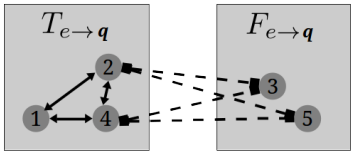


Figure 5.9 – Pairs for data matching deduced from the sets $T_{e\to\boldsymbol{q}}$ and $F_{e\to\boldsymbol{q}}$. In plain arrows are the positive pairs, in dashed lines are the negative pairs.

For each query $\boldsymbol{q}$, we deduce positive and negative pairs M, U for the data matching process from these sets of primary entities $T_{e\to\boldsymbol{q}}$ and $F_{e\to\boldsymbol{q}}$. As described in Figure 5.9, we simply consider that each valid primary entity e for $\boldsymbol{q}$ should be merged to all other valid entities $e' \in T_{e\to\boldsymbol{q}}$, and to none of the invalid entities $e' \in F_{e\to\boldsymbol{q}}$. We do not infer anything about two primary entities that are invalid representant for the query $\boldsymbol{q}$. From this reasoning, we get data-sets M and U by unionizing the positive and negative pairs inferred for each query $q \in \mathcal{Q}$. Formally, the set of positive pairs M and the set of negative pairs U are defined as

$$M = \bigcup_{q\in\mathcal{Q}} \{e, e' \in T_{e\to\boldsymbol{q}} \times T_{e\to\boldsymbol{q}}\} \qquad U = \bigcup_{q\in\mathcal{Q}} \{e, e' \in T_{e\to\boldsymbol{q}} \times F_{e\to\boldsymbol{q}}\} \tag{5.5}$$

where the pairs are counted once when building M. The metrics for the data matching quality (Equation (5.4)) are directly computed from these sets, so that

the system can be tuned based on this implicit feedback on the *merging* process. This tuning can be manual but also automatic, when supervised learning is used for the data matching.

The process we propose is therefore an "entity linking-based" approach to create the labeled data-set for data matching. The major disadvantage of this approach is that the data-set creation directly depends on the entity linking process, meaning we need an entity linking and moreover an efficient one. Indeed, the worst case when $T_{e \to \boldsymbol{q}} = \emptyset$ does not lead to create labeled data-sets M, U. Besides this point, the approach shows several advantages:

1. The similarity between compared primary entities e is indirect, because we "pass by queries" to link the pairs, contrary to [Bilenko and Mooney, 2003] where they use a direct similarity. In our case, the entities e of M do not necessarily have many words in common, but are *conceptually* similar, since they represent the concept behind the query $\boldsymbol{q}$.

2. Our approach cheaply creates a high number of labeled data for the data matching process, with for instance $42,150$ labeled pairs with only $2,709$ manual validations for our case study (see table 5.6 of Section 5.3.5).

3. The set of negative examples U has a relatively similar size of the one of the positive set M, so that the TN value does not bias the evaluation, contrary to what [Christen and Goiser, 2007] observes in usual data matching processes.

4. The computed metrics correspond to an average in the space of entity linking queries $\mathcal{Q}$. This is an advantage because the primary entities are sampled by respecting the distribution of concepts for the final application. Indeed, entities related to queries $\boldsymbol{q}$ which occurs more in real life are more represented in M and U, and the corresponding metrics reflect the importance of the concepts behind each query $\boldsymbol{q}$.

We implemented this evaluation process leveraging the implicit feedback, in order to perform the quantitative comparison described in the following section.

5.3.5 Experimental Results

In this section, we describe the variants of data matching and entity linking we evaluated using the proposed process, and the experimental results obtained.

Dissecting the System

Firstly, we implemented the following variants tested for the *merging* process of Section 5.3.2:

None: This first matching simply aggregates the multiple sources without any merge.

Web: This process merges only the primary entities having the same *website*.

NoLoc: For this data matching, primary entities are merged if they have the same official website, or, if one website is unspecified, if an alias of the one is contained in an alias of the other. These rules are equivalent to those described in Section 5.3.2 without the condition on the institution location.

Full: This matching is the one described in Section 5.3.2.

In Figure 5.5 can be seen the sizes of the sources and the knowledge bases resulting from each data matching. We see that only few primary entities are merged in average, with a noticeable difference among the rules for data matching, with fewer deduplications occurring for stricter rules. Such figures are limited and do not enable us to properly compare the data matching processes.

Knowledge Base	$\mathcal{E}_{\text{LETUD}}$	$\mathcal{E}_{\text{DBP-FR}}$	$\mathcal{E}_{\text{DBP-EN}}$	None	Web	NoLoc	Full
# entities	8,209	3,448	18,639	30,296	27,863	26,491	26,826
% with website	97	94	76	84	82	84	83
Average $\lvert aliases \rvert$	2.19	5.23	4.22	3.78	3.97	5.19	4.04
Average $\lvert locations \rvert$	1.00	1.15	1.81	1.51	1.63	1.74	1.70
Average $\lvert E \rvert$	1.00	1.00	1.00	1.00	1.08	1.14	1.13

Table 5.5 – Statistics for the initial knowledge bases as well as for the knowledge base merged by the variants of the data matching process. The three last lines are quantities averaged per entity.

Similarly, we dissect the *linking* process of Section 5.3.3 by adding incrementally the terms of Equation (5.2) in the score function, in order to measure the features effectiveness:

Semantic: The simplest entity linking limits the score function to the semantic information of the aliases, so that

$$score(q, E) = 5 \times \max_{\substack{\boldsymbol{alias} \\ \in aliases}} cos(\boldsymbol{q}, \boldsymbol{alias}) + cos\Big(\boldsymbol{q}, \bigcup_{\substack{\boldsymbol{alias} \\ \in aliases}} \boldsymbol{alias}\Big)$$

S-Boost: As first variant, the boost on the number of students, $log(students)$, is added to the semantic score.

L-Boost: As second variant, the boost on the location of the institution, $3 \times cos(\boldsymbol{q}, \boldsymbol{locations})$, is added to the semantic score.

All: This last entity linking combines the semantic score with the two boosts (Equation (5.2)).

This change in the formula for $score(\boldsymbol{q}, E)$ also impacts the filtering of relevant entities $Rel(\boldsymbol{q})$ for the *Semantic* and *S-boost* processes, for which the location label is not included in the query $\boldsymbol{q}$.

Labeled Data-sets Created

For each of the entity linking variants, and for every knowledge base resulting from the data matching approaches, we tested the standardization system on real-world queries, following the manual validation process of algorithm 5.3.4. To do so, we extracted the data-set of queries $\mathcal{Q}$ from Viadeo[7] profiles. 263 distinct users profiles were considered, and for each of the 500 educational institutions cited we extracted a query $\boldsymbol{q}$. This forms the set $\mathcal{Q}$ on which the validation process was performed to generate the labeled data-sets Γ_{EL}, M and U for every variant of the system.

Considered set	$\boldsymbol{q} \in \mathcal{Q}$	Validations	$T_{E\to\boldsymbol{q}}$	$F_{E\to\boldsymbol{q}}$	$T_{e\to\boldsymbol{q}}$	$F_{e\to\boldsymbol{q}}$	M	U
Average size	1	5.42	2.45	2.98	4.25	6.86	43.4	40.89
Summed sizes	500	**2709**	1,226	1,483	2,127	3,430	**21,702**	**20,448**

Table 5.6 – Statistics about the validation sets involved in the system evaluation, with quantities averaged per query in the first line. In bold, the number of manual validations, positive pairs and negative pairs for evaluating data matching.

Table 5.6 shows the average size and total size for the intermediate sets, namely $T_{E\to\boldsymbol{q}}$, $F_{E\to\boldsymbol{q}}$, $T_{e\to\boldsymbol{q}}$ and $F_{e\to\boldsymbol{q}}$. The total size for the two first sets informs us that $2,709$ answers of the system have been manually validated, which makes an high average per query: this shows that the tested variants of data matching give significantly different answers for real-life queries. Similarly, the average size of $T_{E\to\boldsymbol{q}}$ is relatively high: this is because for the queries related to major universities, the answers including the corresponding colleges have been accepted, giving thus several variants of E accepted. As an example, this implicitly means the colleges of Oxford should all be merged: this could be discussable, but the data matching being ruled by the final use of standardization, the real world needs influence the ground truth for the data matching. Because of such grouping, the number of labeled pairs for data matching is very large, especially when compared with the number of manual validation. Another positive aspect we observe is that the sets M and U of respectively positive and negative pairs are balanced, which was expected and desired, since we compare primary entities related to the concepts behind the queries.

[7] `http://www.viadeo.com/`

Quantitative Results

DM \ EL	*Semantic*	*S-Boost*	*L-Boost*	*All*
None	0.797	0.802	0.798	0.803
Web	0.802	**0.809**	0.802	**0.809**
NoLoc	0.740	0.740	0.739	0.741
Full	0.800	0.804	0.800	0.804

Table 5.7 – $f - measure_{EL}$ for variants of EL and DM methods. Figures for rec_{EL} or $prec_{EL}$ are similar but skipped for clarity. The variants for the data matching are hard to compare with such figures - especially the Web and Full ones - so that it is impossible to choose one specific merging process.

Table 5.7 shows the results for the two-part system, which corresponds to a classical evaluation of an entity linking process. We see that the data matching highly influences the quality of the two-part system, but the following metrics give no clue about which tuning - like stricter or stronger rules - would improve the knowledge base. Typically, a proper evaluation of the data matching should answer questions like "for which reasons the *NoLoc* variant is worse than the *None* matching?", or "which of the *None*, the *Web* or the *Full* is the best approach for entity linking?", whereas in the previous table the comparison is different for each entity linking. Moreover, the best system seems to be with the *Web* merging procedure, which is actually false if we take into account the knowledge base $\mathcal{E}_{\text{FINAL}}$ quality.

Data Matching	acc_{DM} (%)	$prec_{DM}$ (%)	rec_{DM} (%)
None	0	0	0
Web	72.3	**99.7**	45.2
Full	**75.6**	**99.6**	**51.6**
NoLoc	56.6	58.9	45.5

Table 5.8 – Evaluation Metrics for the Data Matching variants.

In a second time we computed the data matching metrics based on the implicit feedback, which does not require new manual validations. The results shown in table 5.8 explicitly express how efficient each data matching method is. The *Website* and *Full* matchings shows a very good precision, that we require when building this type of system. On the contrary, the *NoLoc* approach gets a lower precision, but a higher recall, whose reason is that it obeys to less strict rules. But a lower precision means more undesirable merges: this is a really bad behaviour for our final use case, which explains why the respective results are lower in table 5.7. Besides that, the *Full* data matching increases by 10% the recall when compared to the *Website* matching: this implies a *better unicity* in the knowledge base, which is significant for the quality of the two-part system. This aspect was not readable in the previous table 5.7, since when looking at

one single profile, standardization might look correct even with duplicated entities in the nomenclature; but when the candidates having graduated from the same institution are associated by our system to two distinct "twins" entities, the statistics about this institution are biased, and we lose the advantages of standardization.

Metrics / EL	$prec_{EL}$	rec_{EL}
Semantic	86.5	74.4
S-Boost	86.9	**74.8**
L-Boost	87.6	73.6
All	**88.1**	74.0
All - DBpedia	82.6	43.8
All - Letudiant	59.6	46.4

Table 5.9 – Evaluation Metrics when varying the entity linking, with the Full data matching knowledge base. The two last lines present results using the DBPedia knowledge base (obtained after the merge of $\mathcal{E}_{DBP\text{-}FR}$ and $\mathcal{E}_{DBP\text{-}EN}$) and the Letudiant knowledge base $\mathcal{E}_{LETUD}$.

Following the previous quantitative results we implemented the system using the *Full* data matching. The performances of the entity linking variants are compared in table 5.9 using the resulting knowledge base of institutions $\mathcal{E}_{\text{FINAL}}$. The recall is lower when considering the *L-Boost* and *All* entity linking because the selection step misses some institutions that have their location unspecified. However, each feature's relevancy is confirmed since the precision is augmented with both boosts, which is a crucial aspect for the industrial use, so that despite a slightly lower $f - measure$, the *All* linking is preferable. We also compared the results when using as a nomenclature the initial knowledge bases separately, namely the one from Letudiant, $\mathcal{E}_{\text{LETUD}}$, and the merge from $\mathcal{E}_{\text{DBP-FR}}$ and $\mathcal{E}_{\text{DBP-EN}}$. The higher precision with DBPedia confirms - as in Section 5.2.4 - that the aliases it provides are more appropriate for standardization. We also see that merging the bases does not only increase the recall but also the precision, reaching 88% instead of 82% and 59% using DBPedia and Letudiant: when generating the nomenclature, the *merging* process produces entities with richer information, which increases the standardization precision.

Despite the misleading results of table 5.7, the final system implemented is with the *Full* merging process, and the *All* linking, that were described in Sections 5.3.2 and 5.3.3. While thousands of candidate profiles are standardized every day by the system, the recall of the merging process (table 5.8) appears improvable, and an approach would be to use machine learning, which leads to the model proposed in the following section.

5.3.6 Perspectives for Active Learning

The experiments has shown the reliability to use the implicit feedback on standardization in order to generate a data-set for evaluating the data-matching. The manual validation process is unified and just requires validating the standard-

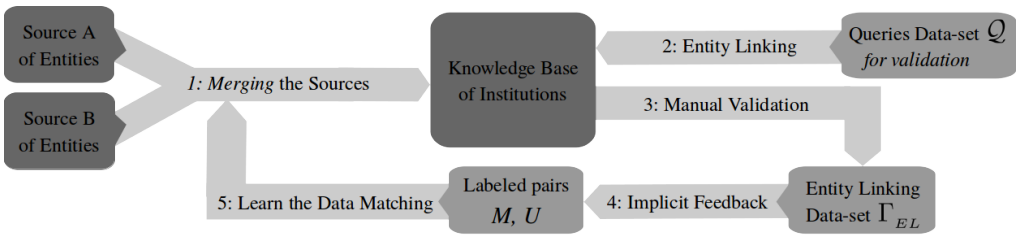


Figure 5.10 – Diagram describing the active learning process. This process iteratively increases the data-sets M, U by repeating the steps 1 to 5.

ization answers, while the resulting metrics are separated into the entity linking and the data matching. In the present study, this process has been used for manually tuning the models parameters, but it could be extended to an automated tuning, in other terms using machine learning. In this case, a classification algorithm is trained on the data-sets of pairs M and U. This classifier then serves at matching more precisely the pairs when merging the sources, while still focusing on the frequent entities, represented by $\mathcal{Q}$. This results in a knowledge base of institutions $\mathcal{E}_{\text{FINAL}}$ of better quality, on which we can perform a new manual validation - by possibly increasing $\mathcal{Q}$ with new real-world queries. This defines an iterating process in which the manual validation keeps increasing the data-sets M, U, which keep improving the data matching and consequently the knowledge base quality.

This system is an active learning, that we introduced in Section 2.3.1. This active learning would apply to any system coupling data matching and entity linking, as illustrated in Figure 5.10. A first advantage is that it would enable us to use machine learning for data matching, which has only been rarely done in the literature due to the difficulty to label pairs, but appeared to be very promising (see Section 2.2.2). A second advantage is that we keep the benefits of the implicit feedback, listed in Section 5.3.4. Besides that, we note that machine learning could also be used for the entity linking, which could therefore also benefit from the active learning, in a more classical way though.

The implementation and in-depth study of this active learning system constitute a future work, that could lead to higher recall for the data matching process (only 51.6% now). However, by pragmatism, we considered the precision and recall of the standardization system as acceptable industrially, so that it has been implemented without studying the active learning process.

5.4 The Limits of External Sources of Knowledge

Through the examples of *Skills* and *Educational Institutions*, this chapter has shown that it is feasible to automatically generate knowledge bases directly usable for standardization. The two systems are now daily used at Multiposting for processing the skills and educational institutions of candidate profiles, as well as the skills of job adverts. The unified approach we developed is first to select a pool of relevant entities from existing knowledge bases, then to merge these

entities, and finally to use this deduplicated knowledge base as a nomenclature.

In each case, the external sources of knowledge have raised some difficulties, so that we had to propose specific methods to build the nomenclature $\mathcal{N}$. Firstly, in the example of the skills, that appears to be concepts hard to define, the selection process was not straightforward but social media helped at selecting the relevant entities for a custom nomenclature. We also have seen through the example of institutions how standardization feedback are helpful for improving the nomenclature construction, and especially the merging process. In the SmartSearch project, the standardization of these two attributes is performed using these generated knowledge bases, which therefore greatly benefits from the public sources of the internet.

Nevertheless, for standardizing some textual fields, there might be no relevant public sources for building a nomenclature $\mathcal{N}$. This is the case of educational degrees, which can be seen as entities but can not be standardized using a comprehensive nomenclature of degrees: there exist very few sources of knowledge about them, and they are far from being comprehensive. In light of this, the approach developed in this chapter does not apply. To discard this issue, we will study in the next chapter an unsupervised learning model (see Section 2.3.2) to generate automatically a labeled data-set from the corpus of profiles. This data-set produced with no external data will constitute, in a sense, our knowledge for standardization, whereas the nomenclature will not be associated to any information.

Chapter 6

Building an Annotated Data-Set for Detecting Study Level

This chapter tackles standardization when it is impossible to build a comprehensive knowledge base of entities like we did in the previous chapter. Instead, we reduce the problem into standardization on an abstract nomenclature, which then relies on pattern recognition on documents (see Section 2.3.1). In such case, we propose to leverage an internal data-set of documents, and apply an unsupervised learning model in order to generate labeled examples, in the form $\boldsymbol{d}, l$ where l is the label to be predicted l and takes values in the abstract nomenclature $\mathcal{N}$. The textual data $\boldsymbol{d}$ represented in this data-set do not constitute a comprehensive overview of the documents to be standardized, and is by no means the standardization nomenclature $\mathcal{N}$. The automatically labeled documents $\boldsymbol{d}, l$ will however be an exploitable knowledge for standardization, from which we will learn how to compute the standardization function $f(\boldsymbol{d})$. This knowledge base makes thus the intermediary between a new document $\boldsymbol{d}$ and its label l, which is the standardized attribute.

With this objective, our object of study will be the degrees of a candidate profile, for which we will fail to leverage any external source of knowledge. Instead of constituting a base of degrees, we focused on detecting the level of training, which is implicit in a degree description. The method we propose only leverages a corpus of candidate profiles - which is internal data - and is trained using on a latent variable model for text, that captures the level of training of a degree description, whereas the other similar models capture the semantic topics (see Section 2.3.2). The comparison with alternative baselines methods, namely state-of-the-art topic models as well as supervised approaches on existing - but limited - annotated data-sets, shows that our unsupervised model is efficient for this problem of text classification. The system has been industrially implemented and is now daily used for standardizing thousands of degrees extracted from candidate profiles.

6.1 Standardizing with no External Knowledge

Our last object of study is the degree, with three examples in Figure 6.1, which includes the Educational **Institution** among other information. Whereas the previous chapter focused purely on the latter, in this chapter we will make use of the institution simply as an enrichment information for our problem, which is the standardization of the degrees themselves.

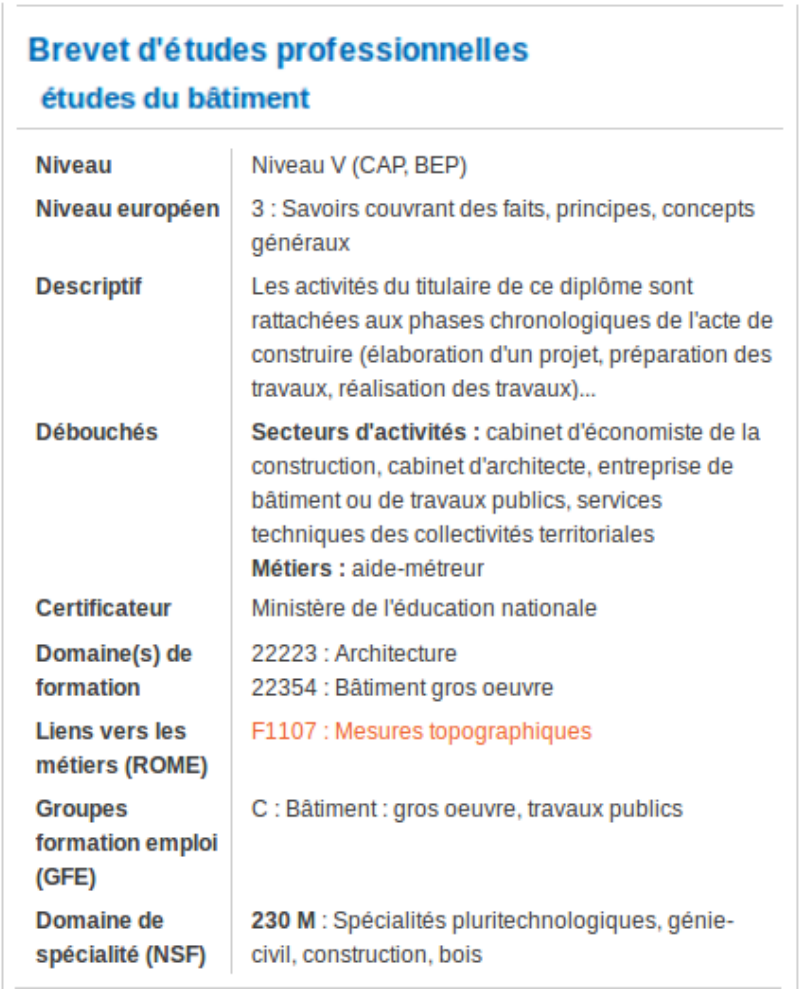
2011-2013	Green University
	Master of Business Administration
2002-2006	Gray College
	B.Eng. in Chemical Industry
1997-2002	Public High School of Springfield
	HS with Honors

Figure 6.1 – Example of an education field content of a candidate, with three degrees. This chapter will not only focus on the educational institution, but on the degree as a whole.

Following to the approach of generating a knowledge base (Chapter 5), a first natural objective for our standardization would be to build a comprehensive base of degrees. This means we consider a degree as an entity, and we standardize it into a nomenclature of entities $\mathcal{N}$, containing possibly hundreds of thousands of degrees for a given country. Like this, standardization would be to link each degree of a resume to its corresponding entry of the degrees knowledge base, from which we would get meta-information such as the level of training, the specialty, or the associated job categories.

With this objective, the first step for building a knowledge base of degrees is the selection of sources of knowledge (Section 5.1.2). Unfortunately, it appears that DBpedia contains only very few degrees among its entities: this is indeed not the purpose of the encyclopedia Wikipedia. Other generic knowledge bases, such as Freebase and Yago, similarly do not contain much data about degrees. There exists however an interesting a domain-specific source called Intercarif-Oref[1], which relates to French education. In this knowledge base, a lot of data is associated to each degree, as shown in Figure 6.2, but Intercarif-Oref has a significant disadvantage: it is not comprehensive and only deals with public and active French degrees. Hence, there is no outdated degrees nor degrees provided by private institutions, which however appear a lot in candidate profiles. Furthermore, in a standardization perspective, the terminology used in Intercarif-Oref appears to be very limited, and do not present any synonyms for instance, which is crucial for a knowledge base serving for standardization (see Section 5.4). This is indeed problematic since the candidates use many abbreviations and aliases when stating their degrees. In light of this, the standardization approach using a knowledge base of degrees appears vain.

[1] `http://www.intercariforef.org`



Brevet d'études professionnelles

études du bâtiment

Niveau	Niveau V (CAP, BEP)
Niveau européen	3 : Savoirs couvrant des faits, principes, concepts généraux
Descriptif	Les activités du titulaire de ce diplôme sont rattachées aux phases chronologiques de l'acte de construire (élaboration d'un projet, préparation des travaux, réalisation des travaux)...
Débouchés	**Secteurs d'activités :** cabinet d'économiste de la construction, cabinet d'architecte, entreprise de bâtiment ou de travaux publics, services techniques des collectivités territoriales **Métiers :** aide-métreur
Certificateur	Ministère de l'éducation nationale
Domaine(s) de formation	22223 : Architecture 22354 : Bâtiment gros oeuvre
Liens vers les métiers (ROME)	F1107 : Mesures topographiques
Groupes formation emploi (GFE)	C : Bâtiment : gros oeuvre, travaux publics
Domaine de spécialité (NSF)	**230 M** : Spécialités pluritechnologiques, génie-civil, construction, bois

Figure 6.2 – Example of data associated to a degree in Intercarif-Oref, such as the level of training, a description, few associated job categories and specialties.

As a consequence, we had to use a different objective for our standardization, which is, in a sense, a simplified objective: instead of seeing the degree as an entity, we propose to detect its level of training, which is crucial for human-resource selection and is a notoriously difficult variable to quantify [Singer and Bruhns, 1991]. This information is not explicitly encoded in the degree name but rather implicit, so that this standardization deals with an abstract nomenclature. The level of training can only be derived by analyzing a large, dynamic and open set of academic institutions, degrees, professional qualifications and corporate certificates. To the best of our knowledge, no satisfactory solution exists to automatically detect this level from multiple unstructured data sources, as those we face in the SmartSearch project.

In term of standardization, the function $f(\boldsymbol{d})$ takes as input a degree description $\boldsymbol{d}$ and takes values in a nomenclature of the levels of training $\mathcal{N}$, that will be later defined in Section 6.2.2. In absence of external knowledge, no data

will be associated to the nomenclature $\mathcal{N}$; our approach is instead to generate some knowledge from a corpus of candidate profiles, that will serve at computing the function f, similarly to the data-set Γ that was used for the job categorization (Chapter 4). With this objective, we will extract the degrees descriptions $\boldsymbol{d}$ contained in the corpus, that express a lot of information such as the outdated degrees and the abbreviations used by candidates. The core of our approach is then to automatically label those documents, which produces a data-set of documents with their weak label:

$$\boldsymbol{d}, l \in \mathcal{D}$$

where $l \in \mathcal{N}$. This data-set $\mathcal{D}$ of weakly labeled documents will serve as knowledge, and contain some mistakes, hence the use of the term "weakly". The standardization process then aims to detect the patterns seen in these examples, which gives the function $f(\boldsymbol{d}) \rightarrow \mathcal{N}$. Contrary to the previous chapter, we thus will not focus on the nomenclature $\mathcal{N}$ but on the function f itself.

In light of this, we present in the following a system articulated in three steps. Firstly, an unsupervised model is trained on a corpus of profiles in order to assign a label to each degree description, leveraging the temporal information through an Expectation Maximization procedure. Secondly, the degrees are extracted from those profiles to form a data-set of weakly labeled degrees. Lastly, a word-based classifier is trained on this data-set, which serves at predicting the training level of a new candidate. The model has been validated on real-world data, showing high accuracy, so that it is now deployed in production in Multiposting products. While the data and the study are located in the French context, the model can easily be adapted to any education-related data.

6.2 Problem Representation and Data

This section starts by a formalization of the educational background of a candidate. The abstract nomenclature $\mathcal{N}$ for the level of training is then described, and followed by a discussion about the temporal aspects of our standardization problem.

6.2.1 Representing the Education of a Candidate

As we focus on the education of a candidate, we will consider for this chapter the education part of a parsed CV, such as the one displayed in Figure 6.1. As stated previously in Section 1.3, this part is a list of degrees, with each an **Institution**, **Degree Description**, **Start date** and **End date**. In light of this, we formalize the educational background b as a sequence of education degree

$$b = b_1, b_2, \cdots, b_n \tag{6.1}$$

where the degrees b_k are in chronological order, the k-th degree b_k (with $k = 1..K$) being defined as:

$$b_k = \boldsymbol{d_k}, sy_k, ey_k \tag{6.2}$$

where $\boldsymbol{d_k}$ represents the k-th degree description, sy_k the corresponding starting date (represented by years) and ey_k the corresponding ending date of the educational program (or the conferring date). To better understand this formalization, the educational background of Figure 6.1 is therefore formalized as:

$$b_1 = \{\text{Public, High, School, ...}\}, 1997, 2002$$
$$b_2 = \{\text{Gray, College, B, Eng}\},\ 2002,\ 2006$$
$$b_3 = \{\text{Green, University, Master, ...}\},\ 2011,\ 2013$$

Where the degree descriptions $\boldsymbol{d_k}$ are truncated for clarity.

6.2.2 The Nomenclature for the Level of Training

Following the formalization, our standardization aims to detect the level of training based on the textual information extracted from the educational background b in a candidate profile. To design a harmonized approach for detecting the level of training, we selected a widely adopted frame of reference: as commonly used in e-recruitment platforms [Faliagka et al., 2012, Sauvageot, 2008], we define the level of training of a degree as the *number of years in tertiary education required to complete it.*

As common basis for the years count, we consider the secondary degree, such as the *Baccalaureate* in France and the *A-levels* in the United Kingdom, which is obviously the necessary requirement for any tertiary degree. Hence, the level of training of a degree can be defined as the number of years of academic training reported by the job candidate after the ending date of the secondary degree. This means that the nomenclature $\mathcal{N}$ of the proposed standardization is constituted of relative integers, ranging from -5 to +8 for our implemented system, where negative values indicate the remaining number of years required to obtain the secondary degree. For the French education, it corresponds to the levels of training "Bac -5", "Bac -4", ... to "Bac +8". This is an abstract nomenclature, since the level of training is never explicitly stated in the degree name.

This formalization of training level has the advantage of being independent from any specific national Education System. However as we aim to harmonize the level of training w.r.t. the secondary degree, we need to consider that the duration of secondary degree can slowly differ between different national education systems. In light of this, the proposed approach is independent from the value of the secondary degree, and aims to maintain the relative order among the degrees, where $\boldsymbol{d} > \boldsymbol{d'}$ means that the degree named $\boldsymbol{d}$ corresponds to a higher training level than the degree named $\boldsymbol{d'}$. The secondary degree can therefore be specified or estimated in our algorithm *a posteriori*, as it will be done in Section 6.3.3. Following this definition of the level of training and the formalization of a candidate, we propose in the next section our strategy to detect the level of a candidate.

Symbol	Description
b	Educational background of a profile, sequence of degrees b_k
b_k	k-th degree of a candidate profile, $\boldsymbol{d_k}$, sy_k, ey_k
k	Index for the degrees of a single candidate (chronological order)
$\boldsymbol{d}$	Degree textual description, *multi-set* of terms d_n
d_n	n-th term for the description $\boldsymbol{d}$, $(n = 1..\|\boldsymbol{d}\|)$, noted $d_{k,n}$ for $\boldsymbol{d_k}$.
sy_k, ey_k	Starting and ending year of the k-th degree b_k
Δy_k	Number of years of academic training after the k first qualifications
l	Level for the degree $\boldsymbol{d}$, indexed by k when it refers to a qualification b_k
l_0	Initial level for an educational background b, at date sy_1
η_b	Probability distribution of level l_0 for the educational background b, verifying $\eta_{b,l_0} = p(l_0\|b)$
$lvl(\boldsymbol{d})$	Term-based prediction of the level of training for a degree description $\boldsymbol{d}$.

Table 6.1 – Notations.

6.2.3 Abandon of Temporal Data for Term-Based Classification

At this point, one notes that the level of training has a purely temporal definition, which might be closely related to the temporal information of a profile, given by the years sy_k and ey_k. An straightforward approach could be to standardize the degrees by using the values of sy_k and ey_k, by counting for instance how many years the candidate has studied to obtain its k-th degree. However, we have to be particularly careful when considering this temporal data: the simple count of the total number of years in education can lead to misleading conclusions. This is due to the following aspects:

1. There might be *gap years* between the degrees. This occurs for example when a person mixes his education with his work experience.

2. The *initial level of training* differs among candidates. By "initial", we mean the level at the date sy_1, which corresponds to the level of the candidate before the first degree stated in his resume. Indeed, some candidates only state their most advanced degree, while others specify all steps from high school.

3. Some candidates' educations presents *temporal irregularities*. This occurs for instance when one switches the field entirely, i.e. first gets a master degree in one area, then gets a master degree in another field: the second master degree requires 2 more years but do not increase the candidate's level. More compromising, some people spend more years to graduate than the normal number of years.

Each of those points are treatable with varying difficulties. The first point is easily solved by counting carefully the years without including the gap years. As an example, with the education of Figure 6.1 we need to deduct the gap between the bachelor and the master. The second point is solvable by estimating the initial level of the candidate, which will be the core of our automatic labeling of Section 6.3. However, even after the estimate of the initial level of a candidate,

to use the temporal data can lead to misleading conclusion because of the last point.

The third point is indeed more compromising since it is unpredictable. Because of these temporal irregularities, for a non negligible number of candidates, the temporal reasoning is not applicable. Consequently, we propose to exclusively *refer to terms for the final prediction*, instead of reasoning on temporal data. In our system, the new candidate profiles will thus be processed based purely on the textual descriptions $\boldsymbol{d_k}$. In light of this, our objective is to build a term-based classifier, written $lvl(\boldsymbol{d})$, which predicts the level of study as defined above from the textual description of the degree $\boldsymbol{d}$. The function $lvl(\boldsymbol{d})$ corresponds to the standardization function - previously written $f(\boldsymbol{d})$ - and takes values in the nomenclature of level of training $\mathcal{N}$. By using this classifier $lvl(\boldsymbol{d})$, we will be able to process a new candidate with education b by assigning him to the maximum detected level among his degrees, each being predicted from the description $\boldsymbol{d_k}$:

$$\text{Level of candidate with background } b = \max_k \ lvl(\boldsymbol{d_k}) \tag{6.3}$$

This way, the final analysis of a candidate is *not* based on the temporal aspect, nor the order of degrees, but only on the words. With the example of Figure 6.1, the predicted levels of training would ideally be

$$\begin{aligned} lvl(\{\text{Public, High, School, ...}\}) &= +0 \\ lvl(\{\text{Gray, College, B, Eng}\}) &= +4 \\ lvl(\{\text{Green, University, Master, ...}\}) &= +6 \\ \Rightarrow \text{Level of the candidate} &= +6 \end{aligned}$$

Building the term-based detector $lvl(\boldsymbol{d})$ is a typical application of textual classification (see Section 2.3.1), provided that we have a data-set of annotated degrees

$$(\boldsymbol{d}, l) \in \mathcal{D} \tag{6.4}$$

As stated during the preliminary study of Section 6.1, it is hard to get a public data-set of this form which is representative and covers properly the terminology used by candidates. In light of this, the data-set $\mathcal{D}$ will be build automatically by leveraging an internal corpus of candidate profiles. The following of this chapter proposes to automatically label the degrees descriptions $\boldsymbol{d}$ contained in an corpus of candidates educational backgrounds, written $\mathcal{B}$. For this labeling - and only for this labeling - we will make use of the sequentiality of the degrees in a profile, as explained in the next section.

6.3 Building a Training Set by Weakly Labeling Candidate Degrees

With the final objective of building a classifier $lvl(\boldsymbol{d})$, we propose to construct automatically a data-set of degrees $\mathcal{D}$ (Equation (6.4)) using live candidate profiles. As it is illustrated in Figures 6.4 and 6.3, the construction takes as input

a data-set $\mathcal{B}$ of candidates' educations background each expressed as b (Section 6.2.1), and is articulated in three steps:

- **Unsupervised labeling** (Section 6.3.2): this step aims to assign a harmonized initial level l_0 to each education b of $\mathcal{B}$, based on the textual and temporal information of b. The estimation is done through a Expectation-Maximization procedure, after the hypothesis of Section 6.3.1.
- **Infer weakly labeled degrees** (Section 6.3.3): this step produces a data-set $\mathcal{D}$ of degrees coupled with their level $(\boldsymbol{d}, l)$. This set is generated using the profiles b of $\mathcal{B}$ and respective estimated initial levels l_0. The levels l expressed in $\mathcal{D}$ are said "weak" because a marginal part is incorrect.
- **Term-based classification training step** (Section 6.3.4): in this step, a supervised classifier $lvl(\boldsymbol{d})$ is trained on the data-set $\mathcal{D}$. This final effective predictor is purely based on the terms and not on the temporal data, and will serve at predicting the level of the new candidates using Equation (6.3).

The next sub-section starts by explaining how the temporal information will be used for generating labels for the degrees contained in the profiles $\mathcal{B}$.

6.3.1 Temporal Aspects

In the objective of building a data-set of labeled degrees, we will automatically assign a label l_k to each degree $\boldsymbol{d_k}$ contained in the education field contents $b \in \mathcal{B}$. To do so, we will leverage the temporal information expressed in each education b. To get rid of the last of the temporal issues listed in Section 6.2.3, the unsupervised labeling step - and only this step - relies on the following assumptions:

1. The candidate increases his/her level of training over time. Each qualification b_j is superior to the previous one b_{j-1}.
2. The candidate completed the every qualification b_k in the normal number of years.

When these assumptions are verified, it is interesting to consider Δy_k, the cumulated number of years of academic training after the k-th degree of the profile b, computed as:

$$\Delta y_k = \sum_{j=1}^{k} ey_j - sy_j$$

where $ey_j - sy_j$ is the number of years used for obtaining the degree b_k. For instance, with the educational background b of Figure 6.1, $\Delta y_1 = 5$, $\Delta y_2 = 9$ and $\Delta y_3 = 11$.

Although both quantities are related, Δy_k is different from the level of the k-th degree. Indeed, by definition, the level of training is with respect to the secondary degree, whereas a candidate provides information about his education

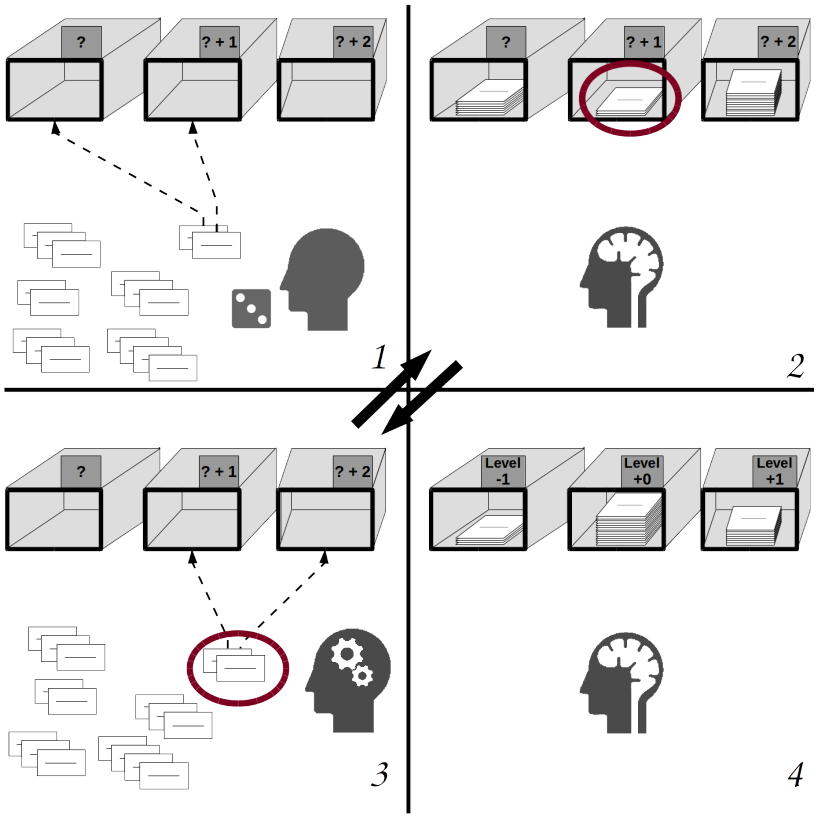


Figure 6.3 – The different steps of our approach as perceived by Bob. In the first frame, he randomly puts the papers in the boxes, using a dice. Since the papers come grouped into sequences, Bob respects the order of the papers when he puts them; the boxes have no clear labels but are ordered (Section 6.3.2). In the second frame, Bob learns the textual patterns appearing in each box, that are filled with papers without any sequential information (Maximization step, Section 6.3.2). After having emptied the boxes, in the third frame, Bob puts again each paper into one box by leveraging these learned patterns as well as the order in each sequence of papers; in particular, the papers' order in a sequence is kept in the ordered boxes (Expectation step, Section 6.3.2). Bob repeats the frames 2 and 3, that is to say the filling and emptying successively the boxes, until the result appears unchanged between two iterations. Then, in frame 4, Bob converts each box into a level of training, by assigning level 0 to the box having the bigger number of documents inside (Section 6.3.3). These boxes filled with papers constitute our knowledge, on which Bob learns the textual patterns for each level of training (Section 6.3.4).

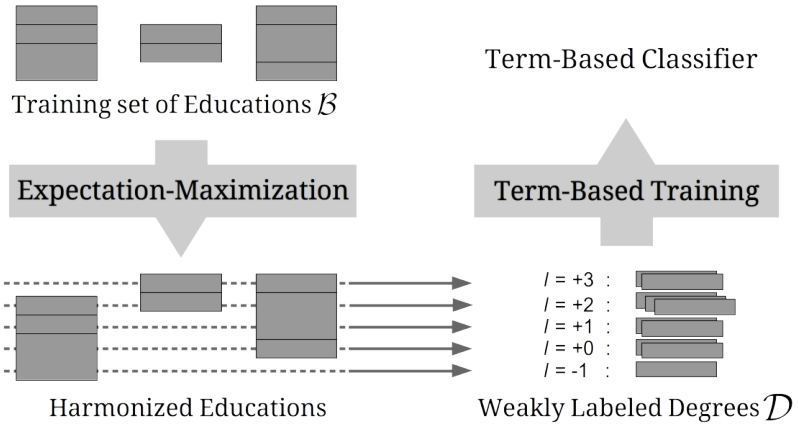


Figure 6.4 – Work flow of the approach.

starting from an arbitrary level. Hence, as stated in Section 6.2.3, we need to consider the initial level of training of an education field b. We write l_0 this value that represents the level of the candidate at the year sy_1. When l_0 is known and the assumptions are verified, each level of training l_k of the k-th degree is given by

$$l_k = l_0 + \Delta y_k \tag{6.5}$$

For instance, the initial level of the education field of Figure 6.1 would ideally be $l_0 = -5$ (i.e.; five years before the secondary degree). The level for b_1, b_2 and b_3 are then given by:

$$l_1 = l_0 + 5 = +0$$
$$l_2 = l_0 + 9 = +4$$
$$l_3 = l_0 + 11 = +6$$

Which are indeed the correct level of training for the respective degrees $\boldsymbol{d_1}$, $\boldsymbol{d_2}$ and $\boldsymbol{d_3}$ (HS / Bachelor / Master).

Since the candidates provide information about their education starting from different levels, the the initial level l_0 takes a different value for each educational background b. Our automatic labelling aims to estimate the value of l_0 for every education b, and associate the label l_k to each degree $\boldsymbol{d_k}$ using Equation ((6.5)). Obviously, for an education not verifying the assumptions above, the equation would give incorrect labels l_k: for this reason, we will say that the resulting labeled degrees are *weakly* labeled. Fortunately, in practice, only a small part of the profiles do not reflect these assumptions.

6.3.2 Unsupervised Labeling of Educational Backgrounds

In this section we propose a method to estimate the value of the initial level l_0 for each educational background $b \in \mathcal{B}$. This value is obviously unknown, but the descriptions $\boldsymbol{d_k}$ observed in b highly depend on it, so that we consider l_0 as a latent variable. After detailing the generative model that links l_0 to the

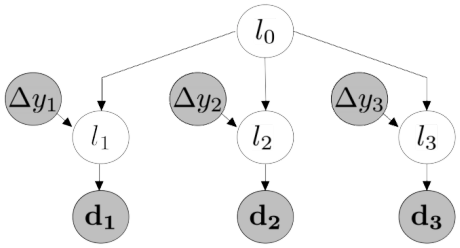


Figure 6.5 – Graphical model for the case of three degrees. Observed variables are in gray.

descriptions $\boldsymbol{d_k}$, we propose an expectation-maximization [Moon, 1996] (EM) procedure for inferring the l_0 values.

Model for the educational background b

In order to motivate that l_0 can be represented as a latent variable, we propose a probabilistic model that generates the descriptions $\boldsymbol{d_k}$ of an educational background b given the value of the initial level l_0 and the temporal information Δy_k. The generative model for an education field b of a candidate profile with K degrees (see Section 6.3.1) is as follows:

- We initially set a level l_0 with probability $p(l_0)$.
- We infer the level of k-th degree by $l_k = l_0 + \Delta y_k$ for $k = 1..K$.
- We generate a description $\boldsymbol{d_k}$ for the k-th degree b_k, term after term, with probability $p(d_{k,n}|l_k)$ (independently from the position k).

In this model, the distribution of the terms of $\boldsymbol{d_k}$ only depends on the value of l_k, and corresponds to a classical naive Bayes distribution among terms (see Section 6.3.2). The distributions $p(d_{k,n}|l_k)$ captures what are the specific terms for each level of training, such as *Master* for $l = +6$ and *Bachelor* for $l = +4$ (American educational system). In the case of an education with 3 degrees, the generative model leads to the following density $p(b, l_0, l_1, l_2, l_3)$, as represented in Figure 6.5:

$$\begin{aligned} p(b, l_0, l_1, l_2, l_3) &= p(\boldsymbol{d_1}, \Delta y_1, \boldsymbol{d_2}, \Delta y_2, \boldsymbol{d_3}, \Delta y_3, l_0, l_1, l_2, l_3) \\ &= p(l_0) \times p(\boldsymbol{d_1}|l_1)p(l_1|l_0, \Delta y_1) \\ &\quad \times p(\boldsymbol{d_2}|l_2)p(l_2|l_0, \Delta y_2) \\ &\quad \times p(\boldsymbol{d_3}|l_3)p(l_3|l_0, \Delta y_3) \end{aligned}$$

At this point, following the assumptions in Section 6.3.1, the level of training of the k-th degree is given by:

$$p(l_k|l_0, \Delta y_k) = \begin{cases} 1 & \text{if } l_k = l_0 + \Delta y_k \\ 0 & \text{otherwise} \end{cases}$$

This dependency among levels l_k, l_0 lets us re-write the probability distribution of a given educational background b as follows:

$$\begin{aligned} p(b, l_0) &= \sum_{l_1,l_2,l_3} p(b, l_0, l_1, l_2, l_3) \\ &= p(l_0) \times p(\boldsymbol{d_1}|l_1 = l_0 + \Delta y_1) \\ &\quad \times p(\boldsymbol{d_2}|l_2 = l_0 + \Delta y_2) \\ &\quad \times p(\boldsymbol{d_3}|l_3 = l_0 + \Delta y_3) \end{aligned}$$

Subsequently, this can be generalized to any number of degrees through this formula:

$$p(b, l_0) = p(l_0) \prod_{k=1}^{K} p(\boldsymbol{d_k}|l_k = l_0 + \Delta y_k) \tag{6.6}$$

where $p(\boldsymbol{d_k}|l_k) = 0$ when $l_k = l_0 + \Delta y_k$ is not in the specified interval of levels of training, which means that l_k is higher than the maximum observed level of training.

The likelihood of the education field b is given by $L(b) = p(b) = \sum_{l_0} p(b, l_0)$, where l_0 covers the interval of level of training, and the total log-likelihood of the corpus $\mathcal{B}$ is then

$$LL(\mathcal{B}) = \sum_{b \in \mathcal{B}} log\left(p\left(b\right)\right) = \sum_{b \in \mathcal{B}} log\left(\sum\nolimits_{l_0} p\left(b, l_0\right)\right)$$

We note that this quantity is intractable for the whole data set $\mathcal{B}$, as the number of possible values for all l_0 grows exponentially with the number of profiles $|\mathcal{B}|$. For this reason, the Expectation-Maximization algorithm instead aims to maximize the expectation of the total log likelihood [Moon, 1996], written $Q(\mathcal{B}) = \sum_{b \in \mathcal{B}} \mathbb{E}_{l_0|b}(log(L(b)))$. To re-write $Q(\mathcal{B})$, for each $b \in \mathcal{B}$ we introduce η_b, the distribution of the initial level l_0:

$$\eta_{b,l_0} = p(l_0|b)$$

Injecting Equation (6.6) into the expression for $Q(\mathcal{B})$ gives:

$$Q(\mathcal{B}) = \sum_{b \in \mathcal{B}} \sum_{\text{level } l_0} \eta_{b,l_0} log\left(p\left(l_0\right)\right) + \sum_{b \in \mathcal{B}} \sum_{\text{level } l_0} \sum_{k} \eta_{b,l_0} log\left(p\left(\boldsymbol{d_k}|l_k\right)\right) \tag{6.7}$$

The Expectation-Maximization procedure maximizes this quantity through an iterative process. After a random initialization, the expectation step and maximization step are repeated until convergence of $Q(\mathcal{B})$. The expectation step updates the distributions η_b of the latent variables l_0, and the maximization step updates the model parameters $p(\boldsymbol{d_k}|l_k)$ and $p(l_0)$. After convergence, every educational background has an estimated distribution η_b for its initial level l_0. These steps are detailed in the next sub-sections; before reading them, it is preferable to be familiar with the EM procedure [Moon, 1996].

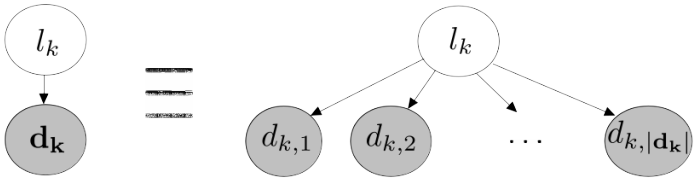


Figure 6.6 – Schema for the generative model of degree descriptions, independent of k. Observed variables are in gray.

Initialization of the initial levels of training

Given an educational background b, the initial level of training l_0 is unknown, so that we first need to assign a random value for starting the EM procedure. To do so, for every education $b \in \mathcal{B}$ we randomly set an integer value for l_0, and we set the distribution η_b such that $\eta_{b,l_0} = 1$ and 0 for all the other components. In our experiments, each l_0 value was sampled uniformly in the range $0, 1, ..., 15$. The upper bound was chosen based on the analysis of the Multiposting profile database, as a maximum training time.

Maximization Step

In this step, for all education b of $\mathcal{B}$, the distribution η_b is fixed, given by the initialization or the expectation of the previous iteration. Given these η_b values, maximizing the expected log-likelihood $Q(\mathcal{B})$ boils down to two independent maximization. Firstly, maximizing the left-hand term of Equation (6.7) gives the distribution of initial level l_0 observed on the educations fields in $\mathcal{B}$:

$$p(l_0) = \frac{\sum_{b \in \mathcal{B}} \eta_{b,l_0}}{|\mathcal{B}|} \tag{6.8}$$

Secondly, maximizing the right-hand term of Equation (6.7) gives the parameters of the model for the degree descriptions given their level $p(\boldsymbol{d_k}|l_k)$. By the change of variable $l_0 = l_k - \Delta y_k$, this second term can be re-written:

$$\sum_{b \in \mathcal{B}} \sum_{k} \sum_{l_k} \eta_{b,l_k - \Delta y_k} log\left(p(\boldsymbol{d_k}|l_k)\right)$$

Given that the documents are generated term by term, this formula corresponds to a simple mixture model [Nigam et al., 1999] where $p(l_k|d_k)$ is replaced by $\eta_{b,l_k-\Delta y_k}$. Indeed, the probability of observing the term $d_{k,n} \in \boldsymbol{d_k}$ for a given level l_k reflects the Naive Bayes assumption, expressed in Figure 6.6 and with the formula:

$$p(\boldsymbol{d_k}|l_k) = p(d_{k,1}, d_{k,2}, ...|l_k) = \prod_{n=1..|\boldsymbol{d_k}|} p(d_{k,n}|l_k) \tag{6.9}$$

Replacing this expression in the previous formula eases at finding the optimal parameters $p(t|l)$. Given a level of training l, the obtained probability to observe

a term t in a degree description is:

$$p(t|l) = \frac{\sum_{b\in\mathcal{B}} \sum_k \sum_n \eta_{b,l-\Delta y_k} \ \mathbb{1}(d_{k,n} = t)}{\sum_{b\in\mathcal{B}} \sum_k \sum_n \eta_{b,l-\Delta y_k}} \tag{6.10}$$

where $\mathbb{1}(d_{k,n} = t)$ equals 1 when $d_{k,n}$ equals the term t, and 0 otherwise. From another point of view, computing the parameters $p(t|l)$ is equivalent to the fuzzy variant of the multinomial Naive Bayes [Nigam et al., 1999], when trained on all the documents $\boldsymbol{d_k}$ with their distribution of label $p(l|\boldsymbol{d_k}) = \eta_{b,l-\Delta y_k}$.

Expectation step

In this step, we fix the model parameters $p(l_0)$ and $p(\boldsymbol{d_k}|l_k)$ that are given by the maximization step of the previous iteration; the expected log-likelihood $Q(\mathcal{B})$ is maximized with respect to the distributions η_b of each educational background b. For each education b and each possible value for the initial level l_0, the probability $p(b, l_0)$ is given by Equation (6.6) used with the previously computed distributions $p(l_0)$ and $p(\boldsymbol{d_k}|l_k)$. A normalization of the $p(b, l_0)$ values leads to the updated distribution η_b:

$$\eta_{b,l_0} = p(l_0|b) = \frac{p(b, l_0)}{\sum_{l'_0} p(b, l'_0)} \tag{6.11}$$

Based on those updated distributions η_b, a new maximization step is performed, and so on until the procedure has converged, as detailed below.

Convergence

The Expectation and Maximization steps are successively iterated. The EM is such that the quantity $Q(\mathcal{B})$ decreases at each iteration. We break the iterative process when this quantity seems to have converged, which is expressed in practice when the variation of $Q(\mathcal{B})$ between two iterations falls below a certain threshold. The whole procedure suffers from the randomness of the initialization (Section 6.3.2), and the final value for $Q(\mathcal{B})$ after convergence is different for every run of the procedure. To reduce this randomness, we perform several runs of the EM and keep the one with the lowest $Q(\mathcal{B})$ value (in our experiments, we performed 5 runs).

After convergence, we have an estimated distribution of initial level η_b for each education $b \in \mathcal{B}$. We detail below how to infer weakly labeled degrees from these aligned profiles.

6.3.3 The Resulting Weakly Labeled Degrees

This step outputs a set of annotated degrees $\mathcal{D}$ containing degree descriptions coupled with their *weak* label, respectively written $\boldsymbol{d}$ and l. Contrary to $\mathcal{B}$, the entries of $\mathcal{D}$ are not sequenced and thus not indexed by k. To build this data-set, from any education $b \in \mathcal{B}$ we extract each degree $\boldsymbol{d_k}$ and estimate a level of training l_k by:

$$l_k = \underset{l_0}{argmax}\ \eta_{b,l_0} + \Delta y_k - \tau \tag{6.12}$$

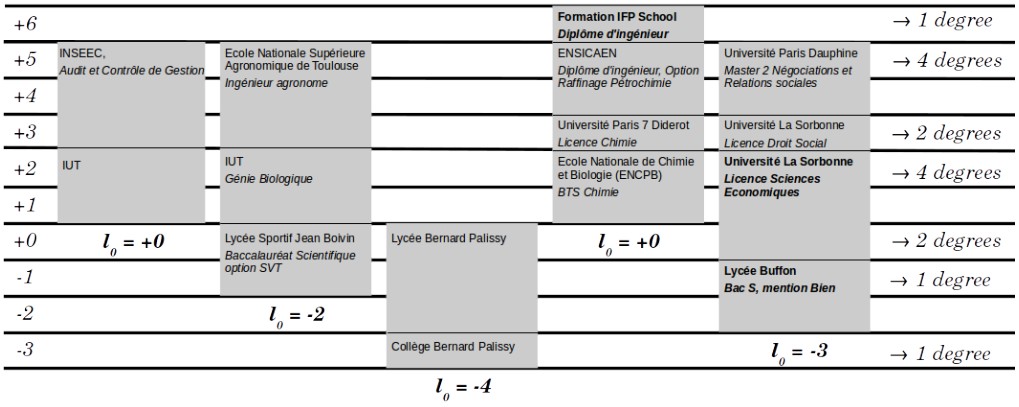


Figure 6.7 – Real case scenario of automatically labeled educations (one background b per column) positioned in the y-axis with respect to their estimated initial level l_0. The degrees of a given row are weakly labeled with the level of training displayed on the left. For the 3 degrees in bold, the weak labels l_k are incorrect, because the corresponding candidates obtained a double degree. However, the term-based classifier resulting from section 6.3.4 gives a correct prediction $lvl(\boldsymbol{d})$.

where $\underset{l_0}{argmax}\ \eta_{b,l_0}$ is the most likely initial level l_0 for the background b, and $\tau \in \mathbb{N}$ is a corrective constant (this corresponds to Equation (6.3.1)). The constant τ serves at ensuring that the l_k values reflect the definition of the level of training, in which the secondary degree has a level set to +0. Indeed, after the automatic labeling, the levels l_k are harmonized between them but take values in $0, ...15$ (subsection 6.3.2) and have no knowledge about the secondary degree.

Theoretically, the secondary degree and thus the value of τ correspond to external information that has to be provided manually. However in our data-set, the secondary degree's level appeared to be with a significant margin the most represented l_0 value in our candidates backgrounds b. In other terms, a large part of the candidates only states the tertiary degrees, so that their education field content starts just after the secondary degree. We thus automatically set

$$\tau = \underset{l_0}{argmax}\ p(l_0)$$

where the distribution $p(l_0)$ is computed on the data-set $\mathcal{B}$ after the automatic labeling. Further experiments are necessary to generalize this property which links the secondary degree with the most represented initial level l_0. The whole labeling process is thus fully automatic; it is summarized in the pseudo-code 2, and some real-world labeled examples are shown in Figure 6.7.

With the example of Figure 6.1, with a proper value for τ the resulting weakly

Algorithm 2: Pseudo-Code for Automatic Labeling

Input: Set $\mathcal{B}$ of educational backgrounds
Output: Set $\mathcal{D}$ of documents $\boldsymbol{d}$ with their label l

for $b \in \mathcal{B}$ **do**
  Randomly assign distribution η_b of l_0 to background b
end
Compute initial $Q(\mathcal{B})$
while $Q(\mathcal{B})$ *has not converged* **do**
  `/* Maximization Step */`
  for *all level of training* l **do**
    Compute terms distribution $p(t|l)$: Equation ((6.10))
  end
  Compute initial level distribution $p(l_0)$: Equation ((6.8))
  `/* Expectation Step */`
  for $b \in \mathcal{B}$ **do**
    Update distribution of $p(l_0|b)$: Equation ((6.11))
  end
  Update $Q(\mathcal{B})$
end
for $b \in \mathcal{B}$, $b_k \in b$ **do**
  Assign weak label l_k: Equation ((6.12))
  Add d_k, l_k to the output set $\mathcal{D}$
end

labeled degrees are:

$$
\begin{aligned}
l &= +0, \quad \boldsymbol{d} = \{\text{Public, High, School, ...}\} \\
l &= +4, \quad \boldsymbol{d} = \{\text{Gray, College, B, Eng}\} \\
l &= +6, \quad \boldsymbol{d} = \{\text{Green, University, Master, Business, ...}\}
\end{aligned}
$$

which constitutes correctly labeled data. This would not be the case if the assumptions stated in Section 6.3.1 were not satisfied; fortunately, in practice, a majority of the profiles satisfies them, so that the weakly labeled degrees constitute *globally* a valid training set for a term-based classifier.

6.3.4 Final Term-Based Classification

This step produces a classifier $lvl(\boldsymbol{d})$ based on the examples of the data-set $\mathcal{D}$. This is a typical application of the numerous classification algorithms for texts (see Section 2.3.1), which aims to find the terms that are relevant or not for a given class. Here, the classes are the levels of training.

In practice, any algorithm of classification can be trained on the data-set $\mathcal{D}$. The classifier we propose in our system is a *Multinomial Naive Bayes*, which logically corresponds to our generative model for documents (Equation (6.9)). With this model, given a degree description $\boldsymbol{d}$, the predicted level $lvl(\boldsymbol{d})$ is given

by:

$$lvl(\boldsymbol{d}) = \underset{l}{argmax}\ p(l|\boldsymbol{d}) = \underset{l}{argmax}\ p(l) \prod_{n=1..|\boldsymbol{d}|} p(d_n|l) \tag{6.13}$$

where the distributions $p(d_n|l)$ and $p(l)$ are computed on $\mathcal{D}$.

In the objective of justifying the use of the Multinomial Naive Bayes, we also experimented a *linear SVM*. It is indeed an popular and efficient model for textual classification [Sebastiani, 2002], and really differs from the previous model since this latter is a discriminative model. Briefly, for any value of the level l, each term t of the vocabulary is associated to a weight $\alpha_{l,t} \in \mathbb{R}$, expressing how relevant t is for the level l. Given a degree description $\boldsymbol{d}$, using the one-versus-all paradigm, the score $\sum_{n=1..|\boldsymbol{d}|} \alpha_{l,d_n} - \beta_l$ is associated to each level of training l, where $\beta_l \in \mathbb{R}$. The final prediction is then given by:

$$lvl(\boldsymbol{d}) = \underset{l}{argmax} \sum_{n=1..|\boldsymbol{d}|} \alpha_{l,d_n} - \beta_l$$

where β_l and $\alpha_{l,t}$ result from an optimisation on the documents $\boldsymbol{d}$ of $\mathcal{D}$ associated to the level l.

The resulting classifier $lvl(\boldsymbol{d})$ is aware of the heterogeneous terminologies used in candidate profiles. In practice, the whole training process detailed in the current section needs to be performed once. In a daily use, only the term-based prediction (Equations (6.3) and (6.13)) is necessary to detect the level of training of the new candidates' profiles.

6.3.5 Comparison with other Latent Variable Models

In this section, we compare the automatic labeling of Section 6.3.2 to the existing latent variable models for text. To gain some generality, we propose to step back from our industrial problem. The general problem would be to annotate documents that are grouped into sequences $(\boldsymbol{d_1}, ..., \boldsymbol{d_K})$. The model implies that documents' labels l_k of a given sequence depends on a latent variable l_0. We moreover assume that some meta data organizes the labels of each sequence, meaning that the distributions of labels l_k can be determined by coupling the meta data with the latent variable l_0. In our case, the meta data is encapsulated in the Δy_k values, and the label values are simply ruled by $l_k = l_0 + \Delta y_k$, but the dependence $p(l_k|l_0, \Delta y_k)$ might be probabilistic.

We first note that our model could be theoretically compared to a Hidden Markov Model. However, comparing the sequence of labels l_k to a Markov Chain is irrelevant, because the sequence only depends on l_0 and is deterministic given this value. Moreover, our model mainly focuses on the distribution of terms by label $p(\boldsymbol{d}|l)$, and relates directly to latent variable models for text [Aggarwal and Zhai, 2012], also called topic models. Contrary to probabilistic Latent Semantic Analysis [Hofmann, 1999] and Latent Dirichlet Allocation [Blei, 2012], our model does not assume a topic variable for each word (denoted as z_n in the referred papers). This assumption comes naturally with the nature of our problem, as each degree description $\boldsymbol{d}$ corresponds to one and only one level of training l,

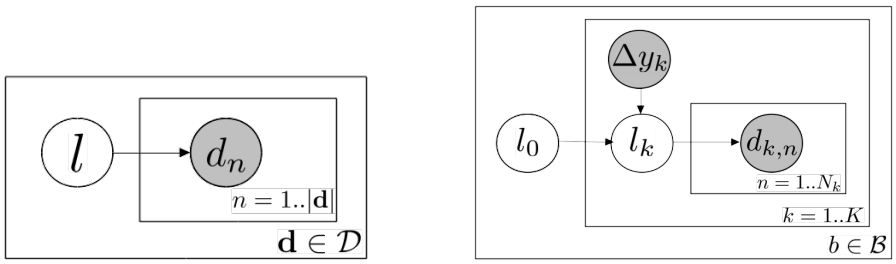


Figure 6.8 – Plate notation for the classical mixture model on the left (which applies to a flat data-set of documents $\mathbf{d} \in \mathcal{D}$) and for our model on the right (which applies to sequences of documents $b \in \mathcal{B}$). Observed variables are in Gray.

so this value is common to all the terms of the document. Our model is thus very related to a mixture of unigrams (modeled in plate notation in the top of Figure 6.8), but goes beyond in two aspects. Firstly, the model incorporates extra temporal information $sy_1, ey_1, ..., sy_K, ey_K$ that organizes the labels to predict, which is made possible because labels are closely related to temporal information (Equation (6.3.1)). Secondly, the documents come as sequences $(\boldsymbol{d_1}, ..., \boldsymbol{d_K})$, for which labels l_k are closely related, leading to a unique distribution per sequence η to estimate in the Expectation step (Section 6.3.2).

6.4 Evaluation and Experiments

This section describes the results of our model when compared to baseline supervised classification methods and latent variable models. With this goal, we describe in detail the alternative data and the metrics used for our evaluation.

6.4.1 Training Data-set

The different steps of Section 6.3 have been performed on a real-word data-set $\mathcal{B}$ of educational backgrounds extracted from SmartSearch database. The profiles have been extracted from Viadeo, one of the most popular and growing professional social network in France. The data-set contains 38,858 candidate profiles with 88,622 different degrees expressed through 29,337 different unique words. Some considered CV examples are shown in Figure 6.7 with their predicted level after the presented labeling step.

For assessing the quality of the resulting classifier $lvl(\boldsymbol{d})$, we predicted the level of the degrees $\boldsymbol{d}$ of the data-sets introduced below.

6.4.2 Alternative Data-sets of Labeled Degrees

In our evaluation we used several labeled data-sets, in which the degrees $\boldsymbol{d}$ come with their level of training l. On the one side, such data-sets can be used to confront the level predictions $lvl(\boldsymbol{d})$ to the ground truth l, whereas the training data-set $\mathcal{B}$ does not provide any ground truth levels. On the other side, each of these data-sets can be used to train a supervised classifier as detailed in Section

	MP-data	IO-data	V-data	$\mathcal{D}$
# degrees	26,312	53,014	500	88,622
# candidates	-	-	217	38,558
# terms	2,319	7,981	426	29,337

Table 6.2 – Statistics for the used data-sets: MP-data, IO-data, V-data and $\mathcal{D}$.

6.3.4, for comparison purposes. In order to meet Multiposting's requirements, the considered data sets are specific to the French education system.[2]

The first considered annotated data-set has been extracted from the base Intercarif-Oref, previously described in section 6.1. The corresponding set of annotated degrees is called *IO-data* and includes a list of annotated *public* and *active* degrees released by public institutions. Considering that many candidates either have degrees released by private institutions or still reports degrees no longer available, we included a complementary data set called *MP-data*, which was manually constructed by Multiposting when including the educational institution into the historic software of the firm. As introduced in Section 1.2.2, this software serves at posting jobs on recruitment websites, meaning that the partners institutions are in a limited number. These two first data-sets include a controlled terminology, and do not provide alternative names and abbreviations for degrees and institutions, often found in professional social networks or e-recruitment platforms. We therefore also considered the data-set *V-data* containing 500 degrees extracted from $\mathcal{B}$ and manually labeled by Multiposting domain experts. This last data-set reflects the most the industrial commitments of our standardization system. Indeed, the degrees of *V-data* express realistically the structure, the heterogeneity and the variable quality of the data encountered in the practical use of the level detection. Moreover, the candidates whose level of training will be predicted by our system come from similar sources to the one of *V-data*. A summary of the data sets is shown in Table 6.2.

6.4.3 Evaluation in Comparison with a Supervised Approach

As first experiment, we evaluated the performances of the classifier trained on $\mathcal{D}$ resulting from our model, against the same classifier when trained on each of the labeled data-sets. The resulting classifiers are then compared to the ground-truth, using each labeled data-set as a test set. In the case where the classifier is trained and tested using the same data-set, a 10-fold cross validation has been performed. For each data-set, we used the two classification algorithms detailed in Section 6.3.4. We emphasize the fact that the compared approaches are deeply different, in the sense that building the data-set $\mathcal{D}$ and the resulting classifier is fully unsupervised. On the contrary, training classifiers on *MP-data* and *V-data* is a typical use of supervised classification.

To compare the different classifiers, the following metrics were computed:

- Error rate E. Firstly, we observed the classification errors with respect to

[2]The French categorization of degrees can be found at `http://www.education.gouv.fr/cid59013/codification-des-formations-et-des-diplomes.html`

the ground truth. The error rate $E \in [0,1]$ is calculated as $1-P$, where P is the classification precision with respect to the predicted level $lvl(\boldsymbol{d})$.

- Hinge-Loss HL. The Hinge-Loss function computes the average distance between the proposed model and the ground-truth data for multi-class categorizations [Moore and DeNero, 2011]. For each prediction, HL considers the vector of the decision function generated by the classifier. This vector is defined as $\hat{\eta}_l = p(l|\boldsymbol{d})\ \forall l$ for the Naive Bayes classifier, and the distance to the hyper-plane for the SVM approaches. These vectors are then compared to the ground truth vector η_l, in which the correct level is marked as 1 (while the others are set to 0):

$$\max_{l}\left(0, 1+\max_{l'\neq l}\hat{\eta}_{l'}-\hat{\eta}_l\right)$$

As shown in Table 6.3 and Table 6.4, our final classifier outperforms, with respect to all of the considered data-set and all the presented metrics, the baseline classifiers obtained after supervised learning. Our system gives indeed satisfactory results for the *MP-data* and *IO-data* data-sets (the best ones after the classifiers trained on degrees of the same data-set), and gives by far the best prediction quality on *V-data*, which constitutes the real-world target data of our standardization system. More specifically, using a Multinomial Naive Bayes, the error rate (0.119) for the introduced method is significantly lower than the ones of the supervised alternatives (0.248 and 0.211 respectively). These performance confirms that the weakly labeled degrees of $\mathcal{D}$ are globally correctly labeled, even if our proposed method to generate $\mathcal{D}$ is fully unsupervised.

Test Set \ Train Set		MP-data	IO-data	$\mathcal{D}$
MP-data	E	***0.070***	0.094	0.089
	HL	***1.647***	1.983	1.923
IO-data	E	0.222	***0.091***	0.154
	HL	1.842	*1.872*	**1.825**
V-data	E	0.248	0.211	**0.119**
	HL	1.166	1.254	**0.993**

Table 6.3 – Results for the comparison with supervised learning using a Multinomial Naive Bayes classification, with best evaluation per line in bold and cross-validation evaluations in italics.

Moreover, it is also possible to remark that the best performances are obtained when the algorithm trained on the data-set $\mathcal{D}$ is the Multinomial Naive Bayes, which is not surprising since the generative model used when building $\mathcal{D}$ obeys to the same assumptions for the words of a document. Considering the supervised learning approach, the SVM gives slightly better performances w.r.t. the error rate E; however this is not the case for the Hinge-Loss HL, which is significantly higher for all the considered tests. This is mainly due to the confidence of the classifier when the prediction is false: such confidence increases the overall HL value, because $\max_{x\neq x_t} y_x$ is decreased and y_{l_t} increased.

Test Set \ Train Set		MP-data	IO-data	$\mathcal{D}$
MP-data	E	***0.063***	0.079	0.127
	HL	*3.810*	3.596	**3.091**
IO-data	E	0.173	***0.071***	0.252
	HL	3.779	*4.206*	**3.537**
V-data	E	0.265	0.204	**0.171**
	HL	2.198	1.762	**1.363**

Table 6.4 – Results for the comparison with supervised learning using a SVM classification, with best evaluation per line in bold and cross-validation evaluations in italics.

6.4.4 Evaluation with other Latent Variables Models

This second evaluation aims to compare the automatic labeling approach of Section 6.3.2 with the well-known latent variable models LDA and mixture model. To do so, these alternative models were trained on the degrees descriptions of the data-set of profiles $\mathcal{B}$, for a number of clusters and topics of 15 (the number of levels in our model), using the implementation of the LDA proposed by [Řehůřek and Sojka, 2010]. Since these models take as input a flat set of documents, we considered separately the degrees $\boldsymbol{d}$ extracted from each $b \in \mathcal{B}$, loosing thus the information related to the sequentiality (contrary to our approach).

We then compared the prediction of clusters/topics against the label prediction $p(l|\boldsymbol{d})$ obtained in our model (Equation (6.9)). For each labeled data-set, every degree $\boldsymbol{d}$ has been assigned to a cluster/topic/label c, that we write $\boldsymbol{d} \in c$, using respectively the mixture model, the LDA and the terms distribution of our model. For this experiments, the labels predicted by our model do not need to be converted to levels of training, and thus take values in 0..15. The underlying idea is to evaluate how each model captures clusters/topics related to the levels of training. Indeed, the most desirable situation is that each cluster/topic represent a level of training, which is the objective of our model but could also be experienced with these two baselines.

With this goal, for each approach we compared the degrees grouped by topic/cluster/label against the groups of degrees having the same ground-truth level, using the mutual information score [Vinh et al., 2010]. This metrics measures the correlation of two clustering approaches, and is calculated in our case as

$$\sum_{c \in Clusters} \sum_{level\ l} p\left(l, \boldsymbol{d} \in c\right) \times log\left(\frac{p\left(l, \boldsymbol{d} \in c\right)}{p\left(l\right) p\left(\boldsymbol{d} \in c\right)}\right)$$

where $\boldsymbol{d} \in c$ denotes the assignment of a degree to a cluster/topic/label, and l is the ground-truth level provided in the annotated data-set on which the above probabilities are computed. We opted for the normalized variant of the mutual information score in order to scale the results in the range of $[0, 1]$. As both models are highly random, we performed 20 evaluations to show average results with their standard deviation.

	LDA	Mixture	Our Approach
MP-data	0.31 ± 0.05	0.62 ± 0.03	**0.77 ± 0.01**
IO-data	0.26 ± 0.07	0.34 ± 0.03	**0.52 ± 0.01**
V-data	0.33 ± 0.05	0.46 ± 0.03	**0.76 ± 0.02**

Table 6.5 – Results for the comparison of our model with the mixture model and LDA. In bold are the best results per line.

Results in Table 6.5 shows that our model clearly presents significant benefits w.r.t. the alternatives. Please also notice that the clusters of terms captured by the LDA do not seem correlated to the level of training, which confirms the idea that our problem is more related to a clustering than to a topic model. We also emphasize the fact that only our model provides interpretable clusters, by leveraging the temporal aspect of the levels of training (Section 6.3.3).

6.5 Future Directions for Unsupervised Processes

This chapter positively addressed the problem of detecting the level of training from heterogeneous degrees' descriptions, expressed in an unstructured manner within user-generated CV. The unsupervised model we propose leverages temporal information in a corpus of candidate profiles and harmonizes the profiles educations, which constitutes a data-set of weakly labeled degrees. This knowledge serves at training a term-based classifier usable for standardization, which outperformed the alternative baselines in our experiments. This model requires no supervision, and can thus be updated at no cost by being trained on a new corpus, for handling new diplomas, or for adapting the standardization to a new country. Testing the model on a corpus of another language remains a future work, such data being not available at time of study.

This study proves that the unsupervised approach shows great advantages, since it requires no supervision, no external data, and can stay close to the real-world terminology, when it is trained on real-world data. But unsupervised learning presents a significant limit: in general, there is no meaning associated to the clusters/topics, which is contradictory with the objective of standardization. However in this example, by considering the temporal meta-data, we obtained an unsupervised model with fixed classes even if the model is updated. These classes are our nomenclature and the system performs an exploitable standardization. As future work, we plan to apply a latent variable model adapted to the current and past work positions of the considered candidates. A first approach would be to again leverage the temporal data, coupled with the job category, so that we would predict how skilled is the candidate in a considered domain. Another interesting study would be to design an unsupervised model that captures the specialty of each degree and each work experience. This would require to consider other meta data, in order to produce meaningful and fixed topics usable as a nomenclature. Determining which meta-data should be considered in such a model remains a interrogation at the moment.

Chapter 7

Practical Applications

In this last chapter, we leave aside the standardization problem and have a look at the results of the previous chapters. Based on the positive experimental evaluations, all our standardization systems have been industrially implemented, each dealing with a special case of standardization. As a global results we obtain a corpus of standardized documents, as illustrated in Figure 7.1. As a first application, the meta-search engines for jobs and for candidates will be the occasion to sum up the standardized attributes inferred throughout the thesis. We will secondly use these attributes as input features for a supervised regression which predicts the attractiveness of a job advert. Thirdly, we will present the interface of the SmartSearch project, that leverages the corpus of standardized documents aggregated from the Internet in order to work out a comprehensive analysis of the job market.

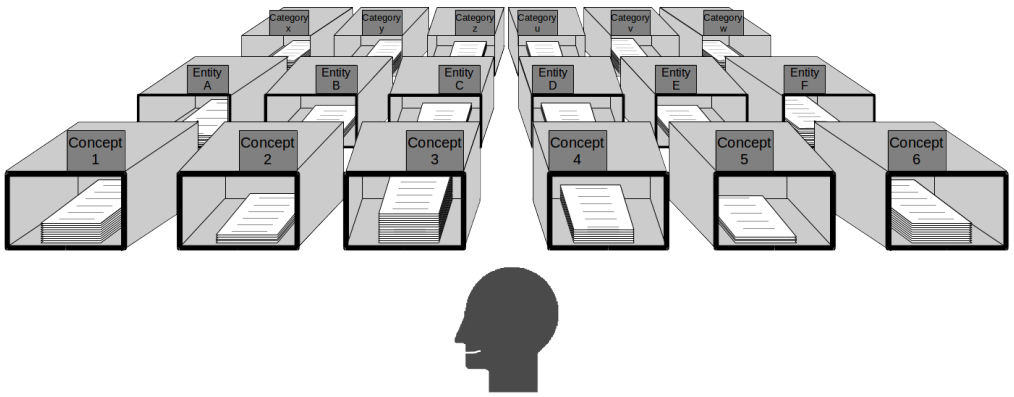


Figure 7.1 – Illustration of Bob's world, now that the documents are standardized. He is smiling: his remaining tasks are the most simple, now that the papers are organized into boxes.

7.1 Meta-Search Engines for Jobs and Candidates

After performing all standardization processes developed in previous chapters, the job adverts and candidate profiles collected in the SmartSearch project are represented with standardized features. Compared to the initial documents, such features represent either existing information but in a standardized way - when this is a nomenclature of entities - either abstract information that was only implicit in the original documents - when this is an abstract nomenclature. In Bob's eyes, at this step, the sheets of papers are all organized into boxes, as illustrated in Figure 7.1. Our work was to ensure that the standardization was meaningful, and now that documents are processed, Bob simply has to retrieve and to count documents. As a first application, the meta-search engine will enable users to specify concept-based queries, that Bob treats by retrieving the documents contained in the corresponding boxes.

We now summarize the documents' standardized attributes that result from our different studies, and on which the meta-search engine users can define structured queries. These attributes are to be compared to the initial documents described in Section 1.3, whose fields were unstructured texts, or structured but only locally. Firstly, the job adverts gained the following attributes thanks to standardization:

Contract, Study, Experience: These attributes have been standardized in Chapter 3, by mapping the websites' nomenclature into a unique nomenclature defined by Multiposting.

Category: The job unstructured fields are used for this standardization into a fine-grained abstract nomenclature (531 categories), as detailed in Chapter 4.

Skills: This list of extracted skills take values in the nomenclature generated in the first part of Chapter 5.

Company, Location: Similarly to the skills, these attributes are standardized using nomenclatures generated following the processes of Section 5.1.2. For brevity worries, these standardization systems are not detailed in this thesis.

As Figure 7.2 shows, these standardized features are used in the queries of the search engine interface. Similarly, on the candidate side, the extracted profiles are now described by the following standardized attributes:

Categories: This is a list of the job categories of each experience of the candidate, obtained as for a job (see above). In most cases, the other attributes of a job advert mentioned above are absent from resumes.

Study level: This is the maximum value of the candidate's degrees levels, each being computed using the model of Chapter 6.

Schools: These educational institutions are standardized using the nomenclature generated in the second part of Chapter 5.

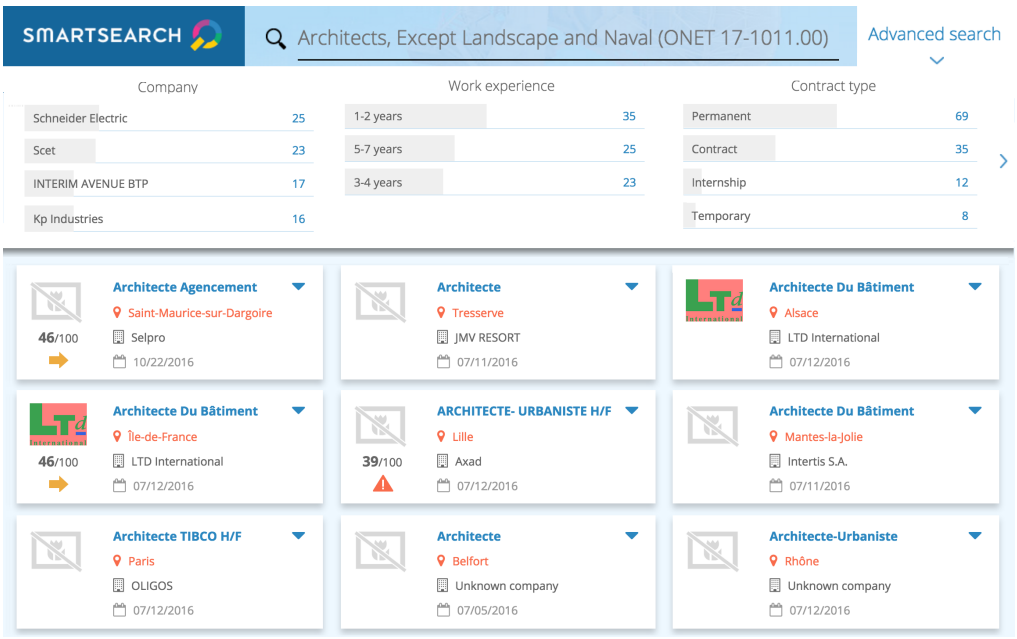


Figure 7.2 – Illustration of jobs search engine. The query is constituted by a job category, on the top of the figure (here, "architect"), and possibly filtered based on other standardized features: company, work experience, contract type, and required study level.

Skills: This attribute has been standardized into the nomenclature generated in the first part of Chapter 5. Since the initial corpus used in this study was extracted also from the profiles of SmartSearch, this standardization might sound biased. This serves however in practice at deduplicating the skills and at filtering those absent in the nomenclature (hence too infrequent, or absent from DBpedia and from social media). The strong advantage of this standardization is to share the same nomenclature than the skills of jobs.

The meta-search engine for candidates is shown in Figure 7.3. Without any standardization, only keyword-based search is possible; it can be performed field by field, thanks to the parsing based on websites' HTML structure (see Figure 1.3 of Section 1.5). To improve ranking, documents have been indexed using some terms weights, as the one introduced in Section 2.1. However, without standardization, the queries remain simply keyword-based. Thanks to standardization, users can search based on meta information, which transforms in a sense the keyword-based search into a concept-based search. Indeed, he can for instance look for a precise entity - such as a university, or a skill - by selecting the right entry in the nomenclature; all the associated documents are then retrieved, including those presenting a synonym of the entity. Moreover, users can look for abstract concepts, for which keyword-based search appears impossible, such as when searching all candidates above a certain level of training. Last but not least, we remind that the search engines presented in Figures 7.3 and 7.3 collect data from all sources displayed previously in Table 1.1 of Section 1.2.2:

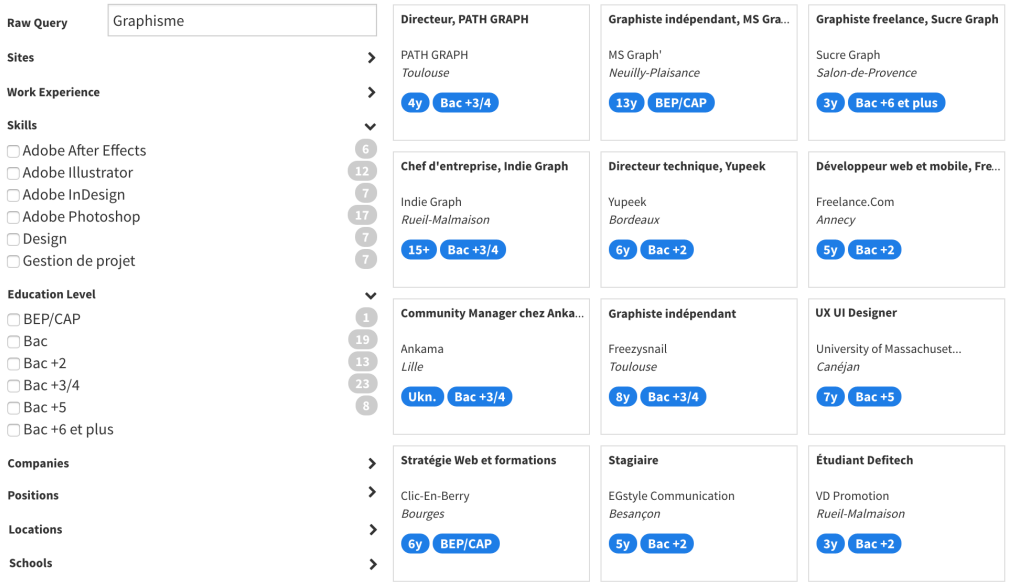


Figure 7.3 – Screenshot of the candidate search engine. The raw query is keyword-based. The other fields are structured, including skills, education level, positions (as job categories) and schools, that each results from the studies of this thesis. Candidates have been here anonymized.

standardization has converted heterogenous data into a common nomenclature.

7.2 Input Features for Predicting Job Attractiveness

Since standardization converts heterogenous data into common features, it can be leveraged to perform some supervised learning on a specific data-set of documents having interesting properties, such as the internal job adverts of Multiposting. Indeed, a major aspect with this internal data-set is that it provides a feedback from the candidates, through their applications. We describe here briefly the notion of attractiveness of an internal job advert of Multiposting, and explain how standardization is used to extend this attractiveness to any job advert of the SmartSearch project.

As previously stated in Sections 1.2.2 and 3.1, the historical tool developed by the firm serves at posting jobs on multiple recruitment websites, called job boards. We get from these postings a feedback through the clicks of applicants: every time a candidate is interested in a job ad, he has to click on the link to apply, meaning that Multiposting can measure the number of clicks for the job j posted on the job board jb:

$$clicks_{jb \to j} \in \mathbb{N}$$

where the value highly depends on the website jb, since certain job boards reach a much wider audience than others. To get a normalized quantity, we leverage the distribution of clicks on the website jb in order to compute what we call the

attractiveness of the job j:

$$att_{jb,j} = p\big(clicks_{jb\to j'} > clicks_{jb\to j}\big) \in [0,1]$$

where the probability is computed on all job adverts j' that have been posted on the job board jb. This quantity compares the performance of j against other job adverts j', and therefore enables us to cope with the job boards' heterogeneity. From this local attractiveness, we deduce an average attractiveness for the job advert j, by

$$att_j = \mathbb{E}_{jb}\big(att_{jb,j}\big)$$

where the expectancy is computed on the job boards where the job j was posted. The highest is this value, the most attractive is the job advert j when compared to the job market, regardless of the website where it was posted. The attractiveness is only computable on the jobs of Multiposting's internal data-set, that we write $\mathcal{J}_{\text{MP}}$. However, thanks to standardization we can also estimate the attractiveness of any external job advert. Indeed, we can use the data-set $\mathcal{J}_{\text{MP}}$ to perform a nearest neighbor regression (see Section 2.3.1). In other words, to get the attractiveness att_j of an external job advert j, we compare it to the job adverts j' of Multiposting $\mathcal{J}_{\text{MP}}$ through:

$$att_j = \frac{1}{\mathcal{Z}} \sum_{\substack{j'\in\mathcal{J}_{\text{MP}} \\ sim_{std}(j,j')>\epsilon}} sim_{std}(j,j') \times att_{j'}$$

where $\mathcal{Z} = \sum_{sim_{std}(j,j')>\epsilon} sim_{std}(j,j')$ is a normalization factor, and the similarity $sim_{std}(j,j')$ between two job adverts j, j' is computed using their standardized attributes. This similarity has been manually tuned by few experts of the firm; its full expression and its evaluation go beyond the scope of this chapter. The important point is that it only uses *standardized attributes*, whereas the tested term-based similarities - as the one introduced in Section 2.1 - gave poor predictions. Standardized attributes are therefore meaningful features for training machine learning algorithms, like the ones introduced in Section 2.3.1. Moreover, this example shows that thanks to the fact that standardization applies to any document, we successfully compare any job advert to the internal job adverts of Multiposting.

7.3 Comprehensive Job Market Analysis

The attractiveness estimate as well as the standardized data-sets of the meta-search engines are both part of the SmartSearch project. This tool considers job adverts and candidate profiles in a macroscopic way, in order to work out comprehensive statistics about the job market. This type of analysis has been only made possible recently, with the growth of e-recruitment and the fact that all documents related to the job market are now available on recruitment websites. However, without any textual standardization, it is hardly possible to infer meaningful information from these millions of jobs and candidate profiles, apart from simple keyword trends.

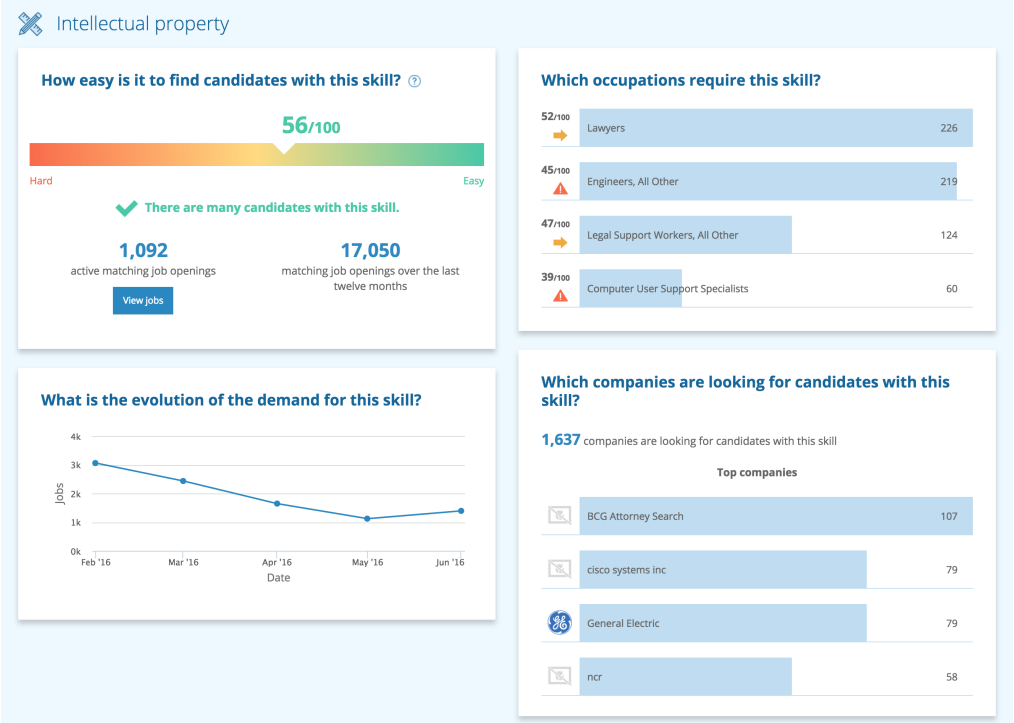


Figure 7.4 – Screenshot of SmartSearch showing statistics about the skill "Intellectual property". On the top left is displayed the averaged attractiveness. On the top right are ranked the most related job categories, and on the bottom right the top hiring companies.

SmartSearch's interface enables the recruiter to specify a concept-based query, which is selected among the nomenclatures of the systems developed in this thesis. As an example, Figure 7.4 shows the statistics of the skill "Intellectual property". The documents having the queried structured attributes form a pool of jobs J, and a pool of candidate profiles C. Some statistics are then computed on these documents, such as the top hiring companies - ranked with respect to the number of corresponding jobs j of J - as well as the average attractiveness of the jobs in J:

$$\frac{1}{|J|}\sum_{j\in J} att_j$$

where each att_j results from the previously described regression, since the job advert o is external and presents no feedback on its attractiveness. One notes that this average is computed on a set of external adverts, which expresses the best the job market, whereas averaging on internal adverts of Multiposting would be biased. Indeed, a kind of job can be under represented in $\mathcal{J}_{\text{MP}}$, but will be properly emphasized when averaging on the pool of external jobs, more representative. The data-set $\mathcal{J}_{\text{MP}}$ serves just at estimating how attractive a job is, which requires only few examples to be meaningful.

Another aspect is that all statistics can be dissected through time, provided that we have historical data-sets, such as in the bottom left of Figure 7.4. Indeed, each job of the data-set of documents is associated to a posting date. One notes that the presented job market analysis only relies on jobs; the interface for candidate-based statistics is under development, and would include information about educational institutions or about skills mastered by employees of a company.

SmartSearch serves thus at answering questions such as "Which skill is popular in this job category?" or "How stable and easy is it to work in this core business?", through the distribution of contract types, attractiveness and required experience/study level. We note however that the computations involve nothing else than counting documents, once standardization has been properly done. From Bob's point of view, his only task for SmartSearch is to count the sheets of paper in the selected boxes.

The different practical applications we presented in this chapter have shown the usefulness of standardization. However, despite these successful practical applications, the use of standardization in an industrial context has highlighted several limitations of our work. In the next chapter, we will draw a global conclusion to our study, before giving future works that would overcome these limitations.

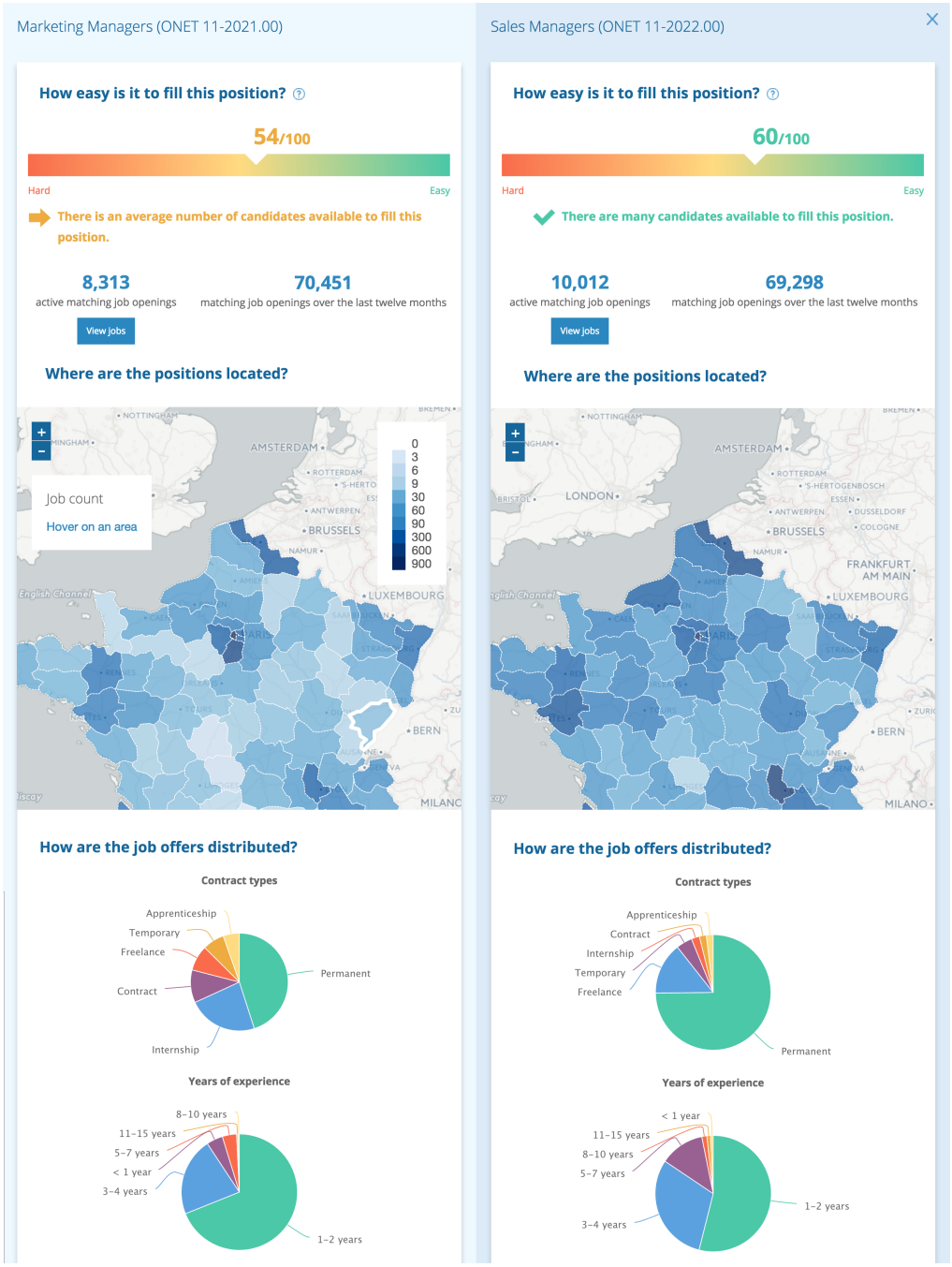


Figure 7.5 – Screenshot of SmartSearch comparing the job categories "Marketing Managers" on the left and "Sales Managers" on the right. The attractivenesses are confronted, along with the distributions of geographical location, contract type and required experience.

Chapter 8

Conclusions

In this thesis we have addressed the problem of textual data standardization, which is raised in the SmartSearch project for processing millions of documents. The first chapter introduced this industrial context and the documents' fields on which we conducted our studies. We also defined the notions of structured textual data, abstract nomenclatures and nomenclatures of entities. The related work studied in Chapter 2 has clarified the two aspects involved in the standardization, namely the nomenclature $\mathcal{N}$ and the standardization function $f(\boldsymbol{d})$. On the nomenclature side, we saw that the Internet present interesting knowledge bases, serving notably at entity linking, but also that existing bases generally do not serve at standardization, because they are too generic or not comprehensive enough. On the standardization function side, we saw that supervised classification well applies to abstract nomenclatures, but requires a large data-set of standardized documents to learn the relevant patterns. Unsupervised learning for text do not need such training data-set, but presents the problem of providing clusters and topics with no meaning, despite the popularity of the latent variable models for text mining.

As a preliminary approach, Chapter 3 has shown that the local nomenclatures are poorly usable for standardization, apart in the case of coarse-grained nomenclatures of entities. For such attribute, the automatic approach we proposed in [Malherbe et al., 2015b] showed good results for mapping nomenclatures, but required a large data-set of manual mappings. This first standardization system introduced further the problem of interoperability that occurs when manipulating documents talking about the same concept but with different words. Moreover, this study showed that reproducing the semantic reasoning is a difficult task, even based on an "ideal" data-set. The size, structure and nature of our data-set was indeed exactly the one of the target data. It appears thus better to handle standardization by considering raw unstructured documents as input and forgetting the knowledge about website. In other words, we recommend not to use the structured knowledge of websites (encoded by the local nomenclatures), but prefer instead to leverage the knowledge encoded by a nomenclature, that would be singular and generic.

Focusing on a generic nomenclature has indeed shown good results in Chapter 4, which confirmed that the structure and richness of $\mathcal{N}$ is at the core of standard-

ization. The chapter addresses the case of a fine-grained abstract nomenclature, and takes as example the job categorization, usually treated by supervised learning on a large labeled data-set. We instead decided to leverage a nomenclature with textual category descriptions, and we proposed in [Malherbe et al., 2014] a learning-to-rank approach that combines the advantages of using the knowledge of $\mathcal{N}$, as well as supervised learning on few standardized examples - that are besides too few to learn a multi-class classifier. We also proposed in [Malherbe et al., 2015a] a semi-supervised approach for enriching the nomenclature $\mathcal{N}$, by adding keywords to the descriptions. In addition, standardization is computed with a probability estimate, that we call correctness. The estimate is based on a supervised learning and applies to any pointwise learning-to-rank system, as we successfully experimented in [Malherbe et al., 2015c]. Using this correctness, we could compare our system with the co-training, a baseline for semi-supervised learning. The results confirmed the relevancy of enriching the nomenclature $\mathcal{N}$ with new keywords instead of increasing the training set for the standardization function f, which is done in co-training.

To face standardization with no pre-defined rich nomenclature, Chapter 5 studied the automated construction of a nomenclature of entities $\mathcal{N}$. For this type of standardization, $\mathcal{N}$ should present rich meta-data like few aliases for each entity. Our work proved that the existing knowledge bases, even if not designed for standardization, can positively be used to generate domain-specific nomenclatures. The generic process we propose involves first selecting the appropriate entities, second merging the sources of knowledge and third linking documents to the corresponding entities. We applied this process successfully to two use cases. Firstly to the skills, as we detailed in [Malherbe and Aufaure, 2016], for which we leveraged the social media and DBpedia. By doing so, the nomenclature is the most up-to-date possible, and reflects the common definition of a skill in a bottom-up construction. Secondly, for treating the educational institutions, we proposed an unified evaluation process, which showed that standardization depends on the merging process quality. Secondly, we saw how the merging quality can inversely be inferred just by evaluating the final standardization. The feedback provided by evaluating the standardized answers can moreover constitute a basis for an active learning for data matching.

Since many situations present no exploitable external knowledge, Chapter 6 addressed the problem of standardization into an abstract nomenclature without using any external source of knowledge. This appeared necessary for the degrees of a candidates, from which we propose to detect the levels of training. This attribute is fully implicit and is very discriminative in e-recruitment. The unsupervised learning process that we successfully proposed in [Malherbe et al., 2016] is based on a novel latent variable model which is trained on an internal corpus of candidate profiles. The model integrates temporal meta-data in order to provide meaningful clusters, and generate a data-set of labeled degrees that constitutes our standardization knowledge. This standardization can thus be automatically updated every time the model is trained on a new corpus of candidate profiles, while keeping the same nomenclature $\mathcal{N}$.

Beyond these satisfactory experimental results, standardization has shown

great applications, as introduced in Chapter 7. The experiments have lead us to implement each of our approaches, that serves at standardizing the millions of documents of SmartSearch. Since all documents are now expressed in the same nomenclatures, it is straightforward to see which concepts are correlated. We can work out concept-based statistics, even if the concepts are entities with many names or abstract concepts that need to be inferred from the documents' patterns. Moreover, this singular structured representation appears to constitute relevant features for supervised learning; for example, after standardizing the internal job adverts of Multiposting, we designed a nearest neighbor regression in order to estimate the attractiveness of a job. This prediction applies to any job advert, and is based on the standardized concepts resulting from our work. We can thus consider a given job sector or industry, look at the corresponding job adverts, and draw statistics including the attractiveness. This feature is just an example of supervised learning, and now that the representative data-set of SmartSearch has been standardized, it opens a way to large-scale machine learning algorithms.

As we formalized it, the standardization problem is generic, and has been applied to many use cases, all around e-recruitment though. In a sense, it provides a data summarization, the working space is reduced to a limited number of values, and it removes the noise inherent to term-based representations. However, we note that each attribute has been treated by different systems, depending on the concepts it deals with - abstract or entities - or on the external data available. We have addressed in this thesis a wide range of cases, that are the ones we encountered in our real-world documents; however, there might be other cases, that would involve a variant of the systems we proposed, or even a deeply different strategy.

8.1 Future Works

A first aspect that should be studied further are the tools and models that can be used in standardization. Indeed, the mathematical tools used in this thesis cover a wide range of fields, with entity linking, topic models, case based reasoning, supervised and semi-supervised classification. In order to improve the presented standardizations or develop new cases of standardization, we should inspect new fields of text analysis. In particular, all models we developed have the limit to be based on the bag-of-words representation of documents. This common assumption for statistical text analysis is indeed very simplifying for long texts with sentences in natural language, such as for Job Descriptions. To take into consideration word order, local grammars [Gross, 1997] in based on an automata which analyze the text word after word. This costly design can be supported by syntactic analysis [Albus et al., 2012], and have been successful in processing jobs [Bsiri et al., 2008], despite an expensive development cost. Another new trend is to use deep neural networks, which are trained on millions of sentences seen as sequences of words [Mikolov et al., 2013a]. Using these networks, each word is represented as a continuous vector, which captures better the semantic concepts [Mikolov et al., 2013b]. This type of network can also be coupled with syntactic

tags [Socher et al., 2013]. The vector can be extended to represent sentences [Le and Mikolov, 2014], which would be of primary interest for standardization. An interesting study would be to adapt sequence to sequence learning [Sutskever et al., 2014] to job adverts, with job description as input sequence and job title as predicted sequence. This would directly give a job categorization, by assigning the input job to the category whose title is the most likely predicted sequence.

A second future work about standardization is a more global prospect and came with the real world usage of SmartSearch. Indeed, our standardization solutions have been treated for given fixed nomenclatures, whereas in practice we would need to adapt to new nomenclatures easily, for instance to the internal nomenclatures of our clients. To reduce the human design work of standardization, we can not afford to build a training set and tune the standardization system for each new nomenclature. An interesting study would therefore be the adaption of standardization to new nomenclatures, based on an existing standardization system. A first natural approach we should use is transfer learning [Pan and Yang, 2010], which could solve the standardization problems sharing the same structure. For instance, this would apply to job categorization in other languages, for which national nomenclatures differs from a country to another but keep a similar field structure. We expect the parameters of the FtFw model to be similar, even with different category descriptions. A more long term strategy would be to build a unified and possibly automatic standardization system. Such system would decide, given a corpus of documents to be standardized, the strategy to adopt. This system would leverage external sources of knowledge, would find the nomenclatures that would fit our problem, and the user would ideally simply confirm wether this appears good. This ambitious perspective would require to study global metrics from the corpus that guides us to one or another strategy.

Another future work will be to leverage further the structured documents resulting from our standardization. Indeed, we now have a very large corpus which presents coherence, with features that has shown to be reliable for supervised prediction in the context of the SmartSearch project. A direction we would like to focus on is to learn embeddings for each concept. This is the focus of deep learning for text, that require a large data-set - that we have, moreover all expressed in the same nomenclatures. However the classical models for embeddings applies to natural language [Mikolov et al., 2013b]; we would thus need to design a novel structure of neural network that would make sense for our documents. These embeddings would lead to a fuzzy representation of our standardized documents, and would in a sense relax the standardization, which appears to be strict - a document is in one box or not. While being discriminative, this single assignment has however permitted to represent simply our strategies through Bob's eyes, who has been quite sympathetic and that we will certainly miss.

Publications

Malherbe, E. and Aufaure, M.-A. (2016). Bridge the terminology gap between recruiters and candidates: A multilingual skills base built from social media and linked data. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*. IEEE.

Malherbe, E., Cataldi, M., and Aufaure, M.-A. (2017). Detect the education level of heterogeneous job candidate profiles: an unsupervised model on cvs for degrees classification. In *Submitted at IEEE International Conference on Data Engineering (ICDE)*.

Malherbe, E., Cataldi, M., and Ballatore, A. (2015a). Bringing order to the job market: Efficient job offer categorization in e-recruitment. *SIGIR '15: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 606151.

Malherbe, E., Diaby, M., Cataldi, M., Viennet, E., and Aufaure, M.-A. (2014). Field selection for job categorization and recommendation to social network users. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 588–595. IEEE.

Malherbe, E., Iwaszko, T., and Aufaure, M.-A. (2015b). A case-based approach for easing schema semantic mapping. In *International Conference on Case-Based Reasoning*, pages 228–243. Springer.

Malherbe, E., Vanrompay, Y., and Aufaure, M.-A. (2015c). From a ranking system to a confidence aware semi-automatic classifier. *Procedia Computer Science*, 60:73–82.

French National Conferences

E. Malherbe and M.-A. Aufaure, "The importance of the correctness of a ranking system: Probability estimation for job categorization," in *Conférence Internationale Francophone sur la Fouille des Données, AAFD & SFC 2016*, May 2016, in Marrakech, Morocco.

References

Aamodt and Plaza, 1994. Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59.

Abbound et al., 2015. Abbound, Y., Boyer, A., and Brun, A. (2015). Predict the emergence: Application to competencies in job offers. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, pages 612–619. IEEE.

Aggarwal and Zhai, 2012. Aggarwal, C. C. and Zhai, C. (2012). *Mining text data*. Springer Science & Business Media.

Albus et al., 2012. Albus, J., Anderson, R., Brayer, J., DeMori, R., Feng, H.-Y., Horowitz, S., Moayer, B., Pavlidis, T., Stallings, W., Swain, P., et al. (2012). *Syntactic pattern recognition, applications*, volume 14. Springer Science & Business Media.

Alterman et al., 2008. Alterman, T., Grosch, J., Chen, X., Chrislip, D., Petersen, M., Krieg Jr, E., Chung, H., and Muntaner, C. (2008). Examining associations between job characteristics and health: linking data from the occupational information network (o* net) to two us national health surveys. *Journal of Occupational and Environmental Medicine*, 50(12):1401–1413.

Andrews et al., 2002. Andrews, S., Tsochantaridis, I., and Hofmann, T. (2002). Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*, pages 561–568.

Archer and Davison, 2008. Archer, W. and Davison, J. (2008). Graduate employability. *The council for industry and Higher Education*.

Arnon and Snider, 2010. Arnon, I. and Snider, N. (2010). More than words: Frequency effects for multi-word phrases. *Journal of memory and language*, 62(1):67–82.

Arora et al., 2012. Arora, S., Ge, R., and Moitra, A. (2012). Learning topic models–going beyond svd. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 1–10. IEEE.

Aumueller and Rahm, 2009. Aumueller, D. and Rahm, E. (2009). Web-based affiliation matching. In *ICIQ*, pages 246–256. Citeseer.

Bellahsene et al., 2011. Bellahsene, Z., Bonifati, A., Rahm, E., et al. (2011). *Schema matching and mapping*, volume 20. Springer.

Berners-Lee et al., 2001. Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5):28–37.

Bernstein et al., 2011. Bernstein, P. A., Madhavan, J., and Rahm, E. (2011). Generic schema matching, ten years later. *Proceedings of the VLDB Endowment*, 4(11):695–701.

Berry and Browne, 2005. Berry, M. W. and Browne, M. (2005). *Understanding search engines: mathematical modeling and text retrieval*, volume 17. Siam.

Bilenko and Mooney, 2003. Bilenko, M. and Mooney, R. J. (2003). On evaluation and training-set construction for duplicate detection. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 7–12.

Bizer et al., 2009a. Bizer, C., Heath, T., and Berners-Lee, T. (2009a). Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227.

Bizer et al., 2009b. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009b). Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165.

Blei, 2004. Blei, D. M. (2004). *Probabilistic models of text and images*. PhD thesis, University of California, Berkeley.

Blei, 2012. Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

Blei et al., 2003. Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Blum and Mitchell, 1998. Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Bouquet et al., 2004. Bouquet, P., Euzenat, J., Franconi, E., Serafini, L., Stamou, G., and Tessaris, S. (2004). D2. 2.1 specification of a common framework for characterizing alignment. In *Peer-to-Peer Knowledge Management, 2004. P2PKM'04. Proceedings. 1st International Workshop on.*

Bourse et al., 2002. Bourse, M., Harzallah, M., Leclère, M., and Trichet, F. (2002). Commoncv: modeling the competencies underlying a curriculum vitae. *Rapport de recherche*, (02.2).

Braun et al., 2010. Braun, S., Kunzmann, C., and Schmidt, A. (2010). People tagging and ontology maturing: Toward collaborative competence management. In *From CSCW to Web 2.0: European Developments in Collaborative Design*, pages 133–154. Springer.

Breiman, 2001. Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Bsiri et al., 2008. Bsiri, S., Geierhos, M., and Ringlstetter, C. (2008). Structuring job search via local grammars. *Advances in Natural Language Processing and Applications. Research in Computing Science (RCS)*, 33:201–212.

Buccafurri et al., 2012. Buccafurri, F., Lax, G., Nocera, A., and Ursino, D. (2012). Discovering hidden me edges in a social internetworking scenario. In *SEBD*, pages 15–26.

Bunescu and Pasca, 2006. Bunescu, R. C. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *EACL*, volume 6, pages 9–16.

Büttcher et al., 2010. Büttcher, S., Clarke, C. L., and Cormack, G. V. (2010). *Information retrieval: Implementing and evaluating search engines*. Mit Press.

Carlson and Schafer, 2008. Carlson, A. and Schafer, C. (2008). Bootstrapping information extraction from semi-structured web pages. In *Machine Learning and Knowledge Discovery in Databases*, pages 195–210. Springer.

Carrizosa and Morales, 2013. Carrizosa, E. and Morales, D. R. (2013). Supervised classification and mathematical optimization. *Computers & Operations Research*, 40(1):150–165.

C.B. Do, 2006. C.B. Do, A. N. (2006). Transfer learning for text classification. *Advances in Neural Information Processing Systems*.

Christen, 2012. Christen, P. (2012). *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media.

Christen and Goiser, 2007. Christen, P. and Goiser, K. (2007). Quality and complexity measures for data linkage and deduplication. In *Quality Measures in Data Mining*, pages 127–151. Springer.

Claveau, 2012. Claveau, V. (2012). Vectorisation, okapi et calcul de similarité pour le tal: pour oublier enfin le tf-idf. In *TALN-Traitement Automatique des Langues Naturelles*.

Cochran, 1953. Cochran, W. G. (1953). *Sampling Techniques*. John Wiley.

Cole et al., 2009. Cole, M. S., Feild, H. S., Giles, W. F., and Harris, S. G. (2009). Recruiters' inferences of applicant personality based on resume screening: Do paper people have a personality? *Journal of Business and Psychology*, 24(1):5–18.

Cole et al., 2007. Cole, M. S., Rubin, R. S., Feild, H. S., and Giles, W. F. (2007). Recruiters' perceptions and use of applicant résumé information: Screening the recent graduate. *Applied Psychology*, 56(2):319–343.

Cornelius et al., 1979. Cornelius, E. T., Carron, T. J., and Collins, M. N. (1979). Job analysis models and job classification. *Personnel Psychology*, 32(4):693–708.

Cornell, 2011. Cornell, J. A. (2011). *Experiments with mixtures: designs, models, and the analysis of mixture data*, volume 895. John Wiley & Sons.

Cover and Hart, 1967. Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

Cucerzan, 2007. Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, volume 7, pages 708–716.

De Smedt et al., 2015. De Smedt, J., le Vrang, M., and Papantoniou, A. (2015). Esco: Towards a semantic web for the european labour market. In *WWW Workshop Linked Data on the Web, Florence, Italy.*

Decker et al., 2000. Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., and Horrocks, I. (2000). The semantic web: The roles of xml and rdf. *Internet Computing, IEEE*, 4(5):63–73.

Deerwester et al., 1990. Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407.

Diaby and Viennet, 2014. Diaby, M. and Viennet, E. (2014). Taxonomy-based job recommender systems on facebook and linkedin profiles. In *Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on*, pages 1–6. IEEE.

Diaby et al., 2013. Diaby, M., Viennet, E., and Launay, T. (2013). Toward the next generation of recruitment tools: An online social network-based job recommender system. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining ASONAM 2013*, pages 821–828.

Dierdorff et al., 2013. Dierdorff, E. C., Norton, J. J., Gregory, C. M., Rivkin, D., and Lewis, P. M. (2013). O*net national perspective on the greening of the world of work. *Green Organizations: Driving Change with I-O Psychology*, pages 348–378.

Do and Rahm, 2002. Do, H.-H. and Rahm, E. (2002). Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 610–621. VLDB Endowment.

Dorn and Naz, 2007. Dorn, J. and Naz, T. (2007). Meta-search in human resource management. In *in Proceedings of 4th International Conference on Knowledge Systems ICKS'07 Bangkok, Thailand*, pages 105–110.

Dumais, 2004. Dumais, S. T. (2004). Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.

Elfeky et al., 2002. Elfeky, M. G., Verykios, V. S., and Elmagarmid, A. K. (2002). Tailor: A record linkage toolbox. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 17–28. IEEE.

et al., 1975. et al., G. S. (1975). A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18(11).

et al., 2004. et al., T.-F. W. (2004). Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*.

Everett, 2013. Everett, B. (2013). *An introduction to latent variable models*. Springer Science & Business Media.

Faliagka et al., 2012. Faliagka, E., Tsakalidis, A., and Tzimas, G. (2012). An integrated e-recruitment system for automated personality mining and applicant ranking. *Internet research*, 22(5):551–568.

Fazel-Zarandi and Fox, 2009. Fazel-Zarandi, M. and Fox, M. S. (2009). Semantic matchmaking for job recruitment: An ontology-based hybrid approach. In *Proceedings of the 8th International Semantic Web Conference*.

Fazel-Zarandi and Fox, 2012. Fazel-Zarandi, M. and Fox, M. S. (2012). An ontology for skill and competency management. In *FOIS*, pages 89–102.

Fellbaum, 2010. Fellbaum, C. (2010). Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.

Fellegi and Sunter, 1969. Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210.

Ghani, 2002. Ghani, R. (2002). Combining labeled and unlabeled data for multiclass text categorization. In *ICML*, volume 2, pages 8–12.

Gomatam et al., 2002. Gomatam, S., Carter, R., Ariet, M., and Mitchell, G. (2002). An empirical comparison of record linkage procedures. *Statistics in medicine*, 21(10):1485–1496.

Gormley and Tong, 2015. Gormley, C. and Tong, Z. (2015). *Elasticsearch: The Definitive Guide*. " O'Reilly Media, Inc.".

Gower and Ross, 1998. Gower, J. C. and Ross, G. J. (1998). Non-probabilistic classification. In *Advances in Data science and Classification*, pages 21–28. Springer.

Gross, 1997. Gross, M. (1997). 11 the construction of local grammars. *Finite-state language processing*, page 329.

Gu and Aamodt, 2006. Gu, M. and Aamodt, A. (2006). Evaluating cbr systems using different data sources: a case study. In *Advances in Case-Based Reasoning*, pages 121–135. Springer.

Hachey et al., 2013. Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. R. (2013). Evaluating entity linking with wikipedia. *Artificial intelligence*, 194:130–150.

Hatami, 2012. Hatami, N. (2012). Thinned-ecoc ensemble based on sequential code shrinking. *Expert Systems with Applications*, 39(1):936–947.

Herzog et al., 2010. Herzog, T. H., Scheuren, F., and Winkler, W. E. (2010). Record linkage. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):535–543.

Hilton et al., 2010. Hilton, M. L., Tippins, N. T., et al. (2010). *A Database for a Changing Economy:: Review of the Occupational Information Network (O* NET)*. National Academies Press.

Hofmann, 1999. Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Hofmann, 2001. Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1-2):177–196.

Hong et al., 2013. Hong, W., Zheng, S., Wang, H., and Shi, J. (2013). A job recommender system based on user clustering. *Journal of Computers*, 8(8):1960–1967.

Hosmer Jr and Lemeshow, 2004. Hosmer Jr, D. W. and Lemeshow, S. (2004). *Applied logistic regression*. John Wiley & Sons.

Hotho et al., 2005. Hotho, A., Nürnberger, A., and Paaß, G. (2005). A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62.

Huan Liu, 2005. Huan Liu, L. Y. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502.

Huang, 2008. Huang, A. (2008). Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56.

Javed et al., 2014. Javed, F., McNair, M., Jacob, F., and Zhao, M. (2014). Towards a job title classification system. *WSCBD 2014 : Web-scale Classification: Classifying Big Data from the Web, WSDM Workshop*.

Ji and Grishman, 2011. Ji, H. and Grishman, R. (2011). Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1148–1158. Association for Computational Linguistics.

Jolliffe, 2002. Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.

Karamatli and Akyokus, 2010. Karamatli, E. and Akyokus, S. (2010). Resume information extraction with named entity clustering based on relationships. In *International Symposium on Innovations in Intelligent Systems and Applications*.

Kim et al., 2011. Kim, D.-k., Motoyama, M., Voelker, G. M., and Saul, L. K. (2011). Topic modeling of freelance job postings to monitor web service abuse. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 11–20. ACM.

Kruppa et al., 2014a. Kruppa, J., Liu, Y., Biau, G., Kohler, M., König, I. R., Malley, J. D., and Ziegler, A. (2014a). Probability estimation with machine learning methods for dichotomous and multicategory outcome: Theory. *Biometrical Journal*, 56(4):534–563.

Kruppa et al., 2014b. Kruppa, J., Liu, Y., Diener, H.-C., Holste, T., Weimar, C., König, I. R., and Ziegler, A. (2014b). Probability estimation with machine learning methods for dichotomous and multicategory outcome: Applications. *Biometrical Journal*, 56(4):564–583.

Landauer et al., 2013. Landauer, T. K., McNamara, D. S., Dennis, S., and Kintsch, W. (2013). *Handbook of latent semantic analysis*. Psychology Press.

Lavallée and Caron, 2001. Lavallée, P. and Caron, P. (2001). Estimation using the generalized weight share method: the case of record linkage. *Survey Methodology*, 27(2):155–169.

Le and Mikolov, 2014. Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Le Ru, 2015. Le Ru, N. (2015). Séries longues d'emploi par métier et par secteur d'activité à partir des enquêtes emploi de l'insee. *Documents d'études de l'INSEE*.

Lewis, 1998. Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer.

Li and Deogun, 2009. Li, D. and Deogun, J. S. (2009). Applications of fuzzy and rough set theory in data mining. In *Methods and Supporting Technologies for Data Analysis*, pages 71–113. Springer.

Litecky et al., 2010. Litecky, C., Aken, A., Ahmad, A., and Nelson, H. J. (2010). Mining for computing jobs. *Software, IEEE*, 27(1):78–85.

Liu et al., 2012. Liu, H., Christiansen, T., Baumgartner Jr, W. A., and Verspoor, K. (2012). Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *J. Biomedical Semantics*, 3(3):17.

Losey, 1998. Losey, M. (1998). Hr comes of age-history of human resource management. *HR Magazine*, 15.

Madhavan et al., 2005a. Madhavan, J., Bernstein, P., Doan, A., Halevy, A., et al. (2005a). Corpus-based schema matching. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 57–68. IEEE.

Madhavan et al., 2005b. Madhavan, J., Bernstein, P., Doan, A., Halevy, A., et al. (2005b). Corpus-based schema matching. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 57–68. IEEE.

Malherbe and Aufaure, 2016. Malherbe, E. and Aufaure, M.-A. (2016). Bridge the terminology gap between recruiters and candidates: A multilingual skills base built from social media and linked data. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*. IEEE.

Malherbe et al., 2016. Malherbe, E., Cataldi, M., and Aufaure, M.-A. (2016). Detecting the level of training of candidates for a job: An unsupervised model for building a weakly labeled data-set from heterogeneous user profiles. In *Submitted at IEEE International Conference on Data Mining (ICDM)*.

Malherbe et al., 2015a. Malherbe, E., Cataldi, M., and Ballatore, A. (2015a). Bringing order to the job market: Efficient job offer categorization in e-recruitment. *SIGIR '15: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 606151.

Malherbe et al., 2014. Malherbe, E., Diaby, M., Cataldi, M., Viennet, E., and Aufaure, M.-A. (2014). Field selection for job categorization and recommendation to social network users. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 588–595. IEEE.

Malherbe et al., 2015b. Malherbe, E., Iwaszko, T., and Aufaure, M.-A. (2015b). A case-based approach for easing schema semantic mapping. In *Case-Based Reasoning Research and Development*, pages 228–243. Springer.

Malherbe et al., 2015c. Malherbe, E., Vanrompay, Y., and Aufaure, M.-A. (2015c). From a ranking system to a confidence aware semi-automatic classifier. *Procedia Computer Science*, 60:73–82.

Manakanatas and Plexousakis, 2006. Manakanatas, D. and Plexousakis, D. (2006). A tool for semi-automated semantic schema mapping: Design and implementation. In *DISWEB*. Citeseer.

Massmann et al., 2011. Massmann, S., Raunich, S., Aumüller, D., Arnold, P., and Rahm, E. (2011). Evolution of the coma match system. *Ontology Matching*, 49.

McCallum, 1999. McCallum, A. (1999). Multi-label text classification with a mixture model trained by em. In *AAAI'99 Workshop on Text Learning*, pages 1–7.

McNamee and Dang, 2009. McNamee, P. and Dang, H. T. (2009). Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113.

Mei and Zhai, 2005. Mei, Q. and Zhai, C. (2005). Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 198–207. ACM.

Mei and Zhai, 2006. Mei, Q. and Zhai, C. (2006). A mixture model for contextual text mining. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 649–655.

Mikolov et al., 2013a. Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov et al., 2013b. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mimno and McCallum, 2008. Mimno, D. and McCallum, A. (2008). Modeling career path trajectories. *Citeseer*.

Mimno and McCallum, 2012. Mimno, D. and McCallum, A. (2012). Topic models conditioned on arbitrary features with dirichlet-multinomial regression. *arXiv preprint arXiv:1206.3278*.

Mochol et al., 2004. Mochol, M., Oldakowski, R., and Heese, R. (2004). Ontology based recruitment process. In *GI Jahrestagung (2)*, pages 198–202.

Moon, 1996. Moon, T. K. (1996). The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60.

Moore and DeNero, 2011. Moore, R. and DeNero, J. (2011). L1 and l2 regularization for multiclass hinge loss models. In *MLSLP*, pages 1–5.

Naz, 2009. Naz, T. (2009). Configurable meta-search in the human resource domain: A hybrid approach to schema and data integration for meta-search engines.

Nie and Fei, 2014. Nie, Qingfeng, L. J. and Fei, S. (2014). Probability estimation for multi-class classification using adaboost. *Pattern Recognition*, 47(12).

Nigam and Ghani, 2000. Nigam, K. and Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM.

Nigam et al., 1999. Nigam, K., Lafferty, J., and McCallum, A. (1999). Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67.

Nigam et al., 2000. Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134.

Omary and Mtenzi, 2010. Omary, Z. and Mtenzi, F. (2010). Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning. *International Journal for Infonomics (IJI)*, 3.

Ouksel and Sheth, 1999. Ouksel, A. M. and Sheth, A. (1999). Semantic interoperability in global information systems. *ACM Sigmod Record*, 28(1):5–12.

Pan and Yang, 2010. Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Pearson, 1894. Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, pages 71–110.

Platt, 1999. Platt, J. C. (1999). Probabilistic output for support vector machines and comparison to regularized likelihood methods. *Advances in large margin classifiers*.

Popescu and Popescu, 2010. Popescu, M. and Popescu, E. (2010). A human resource ontology for recruitment. *Anale. Seria Ştiinţe Economice. Timişoara*, (XVI):896–901.

Porter, 1980. Porter, M. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, 14:130–137.

Porter, 2001. Porter, M. F. (2001). Snowball: A language for stemming algorithms.

Rajaraman et al., 2012. Rajaraman, A., Ullman, J. D., Ullman, J. D., and Ullman, J. D. (2012). *Mining of massive datasets*, volume 77. Cambridge University Press Cambridge.

Rao et al., 2013. Rao, D., McNamee, P., and Dredze, M. (2013). Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, multilingual information extraction and summarization*, pages 93–115. Springer.

Ravikumar and Cohen, 2004. Ravikumar, P. and Cohen, W. W. (2004). A hierarchical graphical model for record linkage. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 454–461. AUAI Press.

Řehůřek and Sojka, 2010. Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Riahi et al., 2012. Riahi, F., Zolaktaf, Z., Shafiei, M., and Milios, E. (2012). Finding expert users in community question answering. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 791–798. ACM.

Robertson and Jones, 1976. Robertson, S. E. and Jones, K. S. (1976). Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146.

Rocchio, 1971. Rocchio, J. (1971). Relevance feedback in information retrieval. *SMART Retrieval System Experimens in Automatic Document Processing*.

Rodrigues et al., 2015. Rodrigues, D., da Silva, A., Rodrigues, R., and dos Santos, E. (2015). Using active learning techniques for improving database schema matching methods. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE.

ROME, 2013. ROME, P. E. (2013). Répertoire opérationnel des métiers et des emplois. *Pôle Emploi. Retrieved from Fiches métiers/emplois ROME: http://www. pole-emploi. fr/candidat/les-fiches-metiers-@/index. jspz*.

Ruthven and Lalmas, 2003. Ruthven, I. and Lalmas, M. (2003). A survey on the us of relevance feedback for information access systems. *The Knowledge Engineering Review*.

Salton et al., 1975. Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Sasaki and Shinnou, 2005. Sasaki, M. and Shinnou, H. (2005). Spam detection using text clustering. In *Cyberworlds, 2005. International Conference on*, pages 4–pp. IEEE.

Sauvageot, 2008. Sauvageot, C. (2008). Un outil au service des comparaisons internationales: la classification internationale type education (cite). *Éducation & formations*, (78):222.

Schmidt et al., 2015. Schmidt, S., Schnitzer, S., and Rensing, C. (2015). Text classification based filters for a domain-specific search engine. *Computers in Industry*.

Schnitzer et al., 2014. Schnitzer, S., Schmidt, S., Rensing, C., and Harriehausen-Mühlbauer, B. (2014). Combining active and ensemble learning for efficient classification of web documents. *Polibits*, (49):39–46.

Sebastiani, 2002. Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Settles, 2010. Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

Shvaiko and Euzenat, 2013a. Shvaiko, P. and Euzenat, J. (2013a). Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):158–176.

Shvaiko and Euzenat, 2013b. Shvaiko, P. and Euzenat, J. (2013b). Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):158–176.

Singer and Bruhns, 1991. Singer, M. S. and Bruhns, C. (1991). Relative effect of applicant work experience and academic qualification on selection interview decisions: A study of between-sample generalizability. *Journal of Applied Psychology*, 76(4):550.

Sivaram and Ramar, 2010. Sivaram, N. and Ramar, K. (2010). Applicability of clustering and classification algorithms for recruitment data mining. *International Journal of Computer Applications*, 4(5):23–28.

Smiley and Pugh, 2011. Smiley, D. and Pugh, E. (2011). *Apache Solr 3 Enterprise Search Server*. Packt Publishing Ltd.

Socher et al., 2013. Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Sugiyama et al., 2003. Sugiyama, K., Hatano, K., Yoshikawa, M., and Uemura, S. (2003). Refinement of tf-idf schemes for web pages using their hyperlinked neighboring pages. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 198–207. ACM.

Sutskever et al., 2014. Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Séguela, 2012. Séguela, J. (2012). *Fouille de données textuelles et systèmes de recommandation appliqués aux offres d'emploi diffusées sur le web*. PhD thesis, Conservatoire National des Arts et Métiers (CNAM), Paris, France.

Séguéla, 2011. Séguéla, J. (2011). Système pour la catégorisation automatique des offres d'emploi en une typologie de fonctions. In *EGC'2011, 11e Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances*, pages 515–526, Brest, France. Prix du meilleur article "Jeune Chercheur".

Tan et al., 1999. Tan, A.-H. et al. (1999). Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, volume 8, page 65.

Tan et al., 2004. Tan, Q., Chai, X., Ng, W., and Lee, D.-L. (2004). Applying co-training to clickthrough data for search engine adaptation. In *Database Systems for Advanced Applications*, pages 519–532. Springer.

Thompson and Willis, 2015. Thompson, C. A. and Willis, C. (2015). Data workforce needs: Disambiguation of roles using clustering and topic modeling. *Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign*.

Trichet and Leclère, 2003. Trichet, F. and Leclère, M. (2003). A framework for building competency-based systems dedicated to human resource management. In *International Symposium on Methodologies for Intelligent Systems*, pages 633–639. Springer.

Tseng, 2010. Tseng, Y.-H. (2010). Generic title labeling for clustered documents. *Expert Systems with Applications*, 37(3):2247–2254.

Vanrompay and Berbers, 2012. Vanrompay, Y. and Berbers, Y. (2012). A methodological approach to quality of future context for proactive smart systems. In Andreev, S. D., Balandin, S., and Koucheryavy, Y., editors, *NEW2AN*, volume 7469 of *Lecture Notes in Computer Science*, pages 152–163. Springer.

Viet et al., 2013. Viet, H. N. Q., Luong, H. X., Miklos, Z., Quan, T. T., and Aberer, K. (2013). Collaborative schema matching reconciliation. In *21st International Conference on Cooperative Information Systems (CoopIS 2013)*.

Vinh et al., 2010. Vinh, N. X., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854.

Wang and Li, 2011. Wang, C. and Li, S. (2011). Corankbayes: Bayesian learning to rank under the co-training framework and its application in keyphrase extraction. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2241–2244. ACM.

Wang and Manning, 2012. Wang, S. and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Watson, 1999. Watson, I. (1999). Case-based reasoning is a methodology not a technology. *Knowledge-based systems*, 12(5):303–308.

Whittaker, 2009. Whittaker, J. (2009). *Graphical models in applied multivariate statistics*. Wiley Publishing.

Winkler, 2006. Winkler, W. E. (2006). Overview of record linkage and current research directions. In *Bureau of the Census*. Citeseer.

Winterton et al., 2009. Winterton, J., Markowitsch, J., and Plaimauer, C. (2009). Descriptors for competence: towards an international standard classification for skills and competences. *Journal of European Industrial Training*, 33(8/9):817–837.

Wu and Weld, 2010. Wu, F. and Weld, D. S. (2010). Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

Zadrozny and Elkan, 2002. Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 26(3):694–699.

Zhang et al., 2013. Zhang, C. J., Chen, L., Jagadish, H., and Cao, C. C. (2013). Reducing uncertainty of schema matching via crowdsourcing. *Proceedings of the VLDB Endowment*, 6(9):757–768.

Zhang et al., 2012. Zhang, S., Li, H., and Zhang, S. (2012). Job opportunity finding by text classification. *Procedia Engineering*, 29:1528–1532.

Zhang et al., 2010. Zhang, W., Su, J., Tan, C. L., and Wang, W. T. (2010). Entity linking leveraging: automatically generated annotation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1290–1298. Association for Computational Linguistics.

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Normalisation Textuelle pour une Analyse Exhaustive du Marché de l'Emploi

Mots clefs : Apprentissage Statistique, Fouille de Textes, Traitement Automatique de la Langue, e-Recrutement

Résumé : Sachant qu'une grande partie des offres d'emplois et des profils candidats est en ligne, le e-recrutement constitue un riche objet d'étude. Ces documents sont cependant des textes non structurés, et le grand nombre ainsi que l'hétérogénéité des sites de recrutement implique une profusion de vocabulaires et nomenclatures. Une difficulté lors de la manipulation de telles données textuelles brutes est d'en déduire les concepts sous-jacents, qui est le problème de normalisation abordé dans cette thèse. Avec l'objectif d'un traitement unifié, la normalisation doit fournir des valeurs dans une nomenclature, de sorte que les attributs résultants forment une représentation structurée unique de l'information. Plusieurs questions se posent alors: peut-on exploiter les structures locales des sites web dans l'objectif d'une normalisation finale unifiée? Quelle structure de nomenclature est la plus adaptée à la normalisation, et comment l'exploiter? Est-il possible de construire automatiquement une telle nomenclature de zéro, ou de normaliser sans en avoir une? Pour illustrer le problème de la normalisation, nous allons étudier par exemple la déduction des compétences ou de la catégorie professionelle d'une offre d'emploi, ou encore du niveau d'étude d'un profil de candidat. Un défi du e-recrutement est que les concepts évoluent continuellement, de sorte que la normalisation se doit de suivre les tendances du marché. A la lumière de cela, nous allons proposer un ensemble de modèles d'apprentissage statistique nécessitant le minimum de supervision et facilement adaptables à l'évolution des nomenclatures. Les questions posées ont trouvé des solutions dans le raisonnement à partir de cas, le learning-to-rank semi-supervisé, les modèles à variable latente, ainsi qu'en bénéficiant de l'Open Data et des médias sociaux. Les différents modèles proposés ont été expérimentés sur des données réelles, avant d'être implémentés industriellement. La normalisation résultante est au coeur de SmartSearch, un projet qui fournit une analyse exhaustive du marché de l'emploi.

Title : Standardization of Textual Data for Comprehensive Job Market Analysis

Keywords : Machine Learning, Text Mining, Natural Language Processing, e-Recruitment

Abstract : With so many job adverts and candidate profiles available online, the e-recruitment constitutes a rich object of study. All this information is however textual data, which from a computational point of view is unstructured. The large number and heterogeneity of recruitment websites also means that there is a lot of vocabularies and nomenclatures. One of the difficulties when dealing with this type of raw textual data is being able to grasp the concepts contained in it, which is the problem of standardization that is tackled in this thesis. The aim of standardization is to create a unified process providing values in a nomenclature. A nomenclature is by definition a finite set of meaningful concepts, which means that the attributes resulting from standardization are a structured representation of the information. Several questions are however raised: Are the websites' structured data usable for a unified standardization? What structure of nomenclature is the best suited for standardization, and how to leverage it? Is it possible to automatically build such a nomenclature from scratch, or to manage the standardization process without one? To illustrate the various obstacles of standardization, the examples we are going to study include the inference of the skills or the category of a job advert, or the level of training of a candidate profile. One of the challenges of e-recruitment is that the concepts are continuously evolving, which means that the standardization must be up-to-date with job market trends. In light of this, we will propose a set of machine learning models that require minimal supervision and can easily adapt to the evolution of the nomenclatures. The questions raised found partial answers using Case Based Reasoning, semi-supervised Learning-to-Rank, latent variable models, and leveraging the evolving sources of the semantic web and social media. The different models proposed have been tested on real-world data, before being implemented in a industrial environment. The resulting standardization is at the core of SmartSearch, a project which provides a comprehensive analysis of the job market.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France