Hierarchical and temporal analysis of scientific corpora as tools for the history of science

Ian Jeantet

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Ian JEANTET

Analyses hiérarchiques et temporelles de corpora scientifiques

vues comme outils pour l'histoire des sciences

Thèse présentée et soutenue à Rennes, le 5 janvier 2021
Unité de recherche : Institut de Recherche en Informatique et Systèmes Aléatoires

Rapporteurs avant soutenance :

Pierre Gançarski	Professeur des universités à l'Université de Strasbourg
Christine Largeron	Professeure des universités à Université Jean Monnet de Saint-Étienne

Composition du Jury :

Président :	Elisa Fromont	Professeure des universités à l'Université de Rennes 1
Examinateurs :	David Chavalarias	Directeur de recherche à l'Institut des Systèmes Complexes - PIF
	Pierre Gançarski	Professeur des universités à l'Université de Strasbourg
	Christine Largeron	Professeure des universités à Université Jean Monnet de Saint-Étienne
	Zoltan Miklos	Maître de conférences à l'Université de Rennes 1
Dir. de thèse :	David Gross-Amblard	Professeur des universités à l'Université de Rennes 1

Résumé

La philosophie des sciences est une branche de l'épistémologie qui s'intéresse aux fondements, méthodes et implications de la science. Les philosophes essaient de définir, d'expliquer et de justifier la science d'une manière philosophique, c'est-à-dire distinguer la science de la non-science, déterminer son but, définir une bonne explication ou une bonne justification. L'histoire des sciences est l'étude du développement de la science. Les historiens des sciences se sont consacrés à répondre à des questions sur ce qu'est la science, comment elle fonctionne et si elle présente des modèles et des tendances visible à grande échelle. Ce sont deux domaines étroitement liés qui fonctionnent souvent ensemble.

Historiquement, il existe trois grands modèles d'évolution et d'interaction entre une ou plusieurs théories scientifiques.

Le premier modèle, associé à Popper, considère que le progrès scientifique passe par une falsification de théories incorrectes et l'adoption à la place de théories qui se rapprochent progressivement de la vérité. La science évolue de manière linéaire par accumulation de faits. La physique d'Aristote était surclassé par la mécanique classique de Newton elle-même surclassé par la relativité d'Einstein, etc.

Le principal modèle challengeant celui de Popper et proposé par Kuhn ne voit pas la science comme un progrès linéaire mais séparée en cycles à deux phases, une phase de science «normale» et une phase de science «révolutionnaire». Si la première phase est une période calme évoluant de manière similaire à la vision de Popper, la seconde, dans laquelle de nouvelles théories basées sur de nouvelles hypothèses sont testées, marque la séparation entre paradigmes. Les paradigmes sont des structures dominantes de pensée et de pratiques et représentent des hypothèses entièrement différentes et incommensurables (non comparables) sur l'univers. Il ne reste que peu de chose du paradigme précédent lorsqu'un changement se produit.

Un autre modèle extrême, promu par Feyerabend, réfute l'existence de méthodologies cohérentes utilisées par tous les scientifiques à tout moment et a argumenté durement contre l'idée que la falsification a jamais été vraiment suivie dans l'histoire de la science. Il a défendu sa position en remarquant qu'en pratique, les scientifiques considèrent arbitrairement les théories comme exactes même si elles ont échoué à de nombreux tests.

Il en va de même pour certaines formes de connaissances considérées tour à tour comme science et non-science.

De nombreuses autres théories ont été proposées pour nuancer ces trois principaux modèles. Ils présentent différentes variations dans la représentation du changement scientifique, de la notion de progrès et de la relation entre théorie et vérité.

Quels que soient les moteurs d'évolution de la science, la méthode scientifique qui produit la science de nos jours nécessite de confronter les théories à la réalité par une validation empirique. Ces résultats sont présentés à la communauté dans des publications évaluées par des pairs. L'analyse des articles scientifiques produits au sein d'un domaine apparaît donc comme un bon moyen de déterminer les théories en jeu et l'étude de l'évolution de ces publications pourrait donner un aperçu des processus d'évolution en jeu. Les pionniers de l'épistémologie quantitative ont essayé de fournir des outils automatiques pour cartographier le paysage scientifique afin de permettre aux historiens des sciences de comprendre pleinement l'évolution de la science. Certains ont tenté de construire des cartes basées sur un index de citation des articles scientifiques et des travaux plus récents ont introduit les notions de phylomémie, carte de métro et flux de texte tentant de générer des structures l'évolution de la science en se basant directement sur une étude du contenu des articles. L'étude du texte en lui-même a permis une granularité plus fine dans les cartes produites. La récente promotion pour une science ouverte a conduit à la disponibilité de très grands corpus de publications scientifique ce qui nécessitent des méthodes d'exploration de données de pointe pour en extraire des modèles d'évolution.

Pour reconstruire automatiquement des structures évolutives, les scientifiques doivent combiner plusieurs outils de traitement du langage naturel et d'exploration de texte suivant une méthodologie divisée en plusieurs étapes et pouvant être définie comme suit :

— Récupération de données et répartition du corpus par périodes. Les périodes typiques pour étudier l'évolution récente de la science sont de quelques années, entre 3 et 5, avec une fenêtre glissante sur les périodes pour avoir un chevauchement des données. Comme la science évolue continuellement, diviser les données par périodes est arbitraire et avoir une division au milieu d'un événement particulier pourrait le rendre non détecté. Le chevauchement est là pour empêcher un tel comportement et identifier un maximum de ces événements.

— Extraction de termes et matrice de similarité. Cette étape transforme l'ensemble des documents de chaque période en une matrice de termes pondérée. Cette matrice vise à représenter la relation entre les termes clés, généralement des n-grammes.

Une représentation classique de cette relation est basée sur la co-occurrence des mots dans un document.

— Détection des thématiques. Cela consiste à détecter des ensembles de termes fortement liés sémantiquement dans des graphes de termes, la contrepartie graphique des matrices de co-occurrence.

— Correspondance inter-temporelle. Cela correspond à la mise en relation de domaines de différentes périodes, généralement en utilisant l'index de Jaccard, pour en générer des lignes d'évolution temporelle. Cela conduira à des scissions dans les structures, similaires à celles présentes dans une structure phylogénétique, mais aussi à des fusions et d'autres événements spécifiques à ce type de carte évolutive. En général, une interaction avec la carte permet aux experts de personnaliser sa génération en modifiant les données (par exemple en supprimant ou en ajoutant un terme dans un domaine) ou les paramètres (par exemple en changeant l'intervalle de temps pour diviser le corpus de documents).

Les travaux présentés dans cette thèse visent à fournir une analyse automatique des publications scientifiques pour l'épistémologie quantitative. Comme plusieurs autres méthodes, l'objectif final est de produire des cartes d'évolution des domaines scientifiques pour aider les épistémologues à déterminer les mécanismes en jeu et ce à partir du texte brut des publications.

Nous proposons d'abord d'enrichir les connaissances sur les relations existantes entre domaines scientifiques avec la définition d'une nouvelle structure hiérarchique appelée quasi-dendrogramme. Cette structure, une généralisation du simple arbre, peut être vue comme un graphe acyclique dirigé spécifique. Nous proposons un cadre d'étude comprenant un nouvel algorithme de regroupement hiérarchique chevauchant (OHC) afin de générer une telle hiérarchie à partir d'un ensemble d'articles scientifiques. De cette façon, à chaque niveau, nous représentons un ensemble de regroupements éventuellement superposés. Si les groupes présents dans les données ne montrent aucun chevauchement, ils sont alors identiques à des regroupements que nous pouvons déterminer en utilisant des méthodes de regroupement agglomératives classiques. Cependant, en cas de chevauchement et de groupes imbriqués notre méthode aboutit à une représentation plus riche qui peut contenir des informations pertinentes sur la structure entre ces groupes. En fait, cet algorithme peut être appliqué à tout autre type de données sur lequel une mesure de similarité est définie, exactement comme pour toute méthode de regroupement agglomérative. Par construction, nous avons prouvé des propriétés théoriques intrinsèques

aux quasi-dendrogrammes et montré comment l'algorithme OHC se comporte sur des données synthétiques et réelles, soulevant le problème de passage à l'échelle inhérent à ce type d'algorithme.

L'un des problèmes majeurs était également l'absence de jeu de données contenant une vérité terrain sur la structure de la science. Nous nous sommes retrouvés dans l'incapacité de comparer les hiérarchies que nous avons produites à de tels standards. Nous avons plutôt opté pour la définition d'une mesure de similarité afin de pouvoir comparer les quasi-dendrogrammes entre eux mais aussi avec d'autres structures hiérarchiques classiques comme les simples dendrogrammes. Nous voulions pouvoir mettre en évidence les spécificités et les régularités qui pouvaient se produire dans un ensemble de hiérarchies générées avec des algorithmes différents ou avec des paramètres différents sur un même jeu de données. À notre connaissance, très peu de travaux ont été effectués dans ce sens dans la littérature scientifique lorsque la structure devient plus complexe qu'un simple arbre. Nous avons donc proposé une nouvelle mesure de similarité qui aborde le problème par niveau. En effet, si pour deux structures hiérarchiques lorsque nous extrayons une couverture de k regroupements nous obtenons des groupes similaires pour toute valeur de k cela signifie que les structures sont très similaires et inversement. Nous avons utilisé cette mesure de similarité pour confirmer par l'expérience que l'algorithme OHC agit comme un algorithme à liaison simple classique lorsque le critère de fusion est proche de 1 et diffère de plus en plus lorsqu'il diminue. Nous avons également proposé une vue détaillée de la similarité pour détecter les parties communes et différentes des hiérarchies et une variante pondérée dans le but de réduire l'impact causé par les petits groupes agissant comme du bruit qui persisteraient dans les haut niveaux des hiérarchies. En fait, la détection de parties communes à plusieurs hiérarchies, c'est-à-dire qui sont produites pour plusieurs critères de fusion, peut être considérée comme un critère de robustesse.

Enfin, nous proposons une méthode alternative pour générer des cartes évolutives de domaines scientifiques à partir d'une requête d'un utilisateur sous la forme d'ensembles de termes représentant des sujets que l'utilisateur souhaite suivre. Dans notre cadre d'étude, une carte évolutive est définie comme un ensemble de chronologies, une chronologie étant une séquence de plusieurs groupes appartenant à des hiérarchies de périodes consécutives. Elles sont générées en utilisant des matrices d'alignement entre les hiérarchies. Nous proposons en détail une méthodologie complète permettant cette reconstruction et appliquons notre proposition à des quasi-dendrogrammes alignés à l'aide d'un algorithme spécifique. Nous avons également défini une probabilité d'évolution qui, si elle est utilisée comme

un seuil, produit des cartes évolutives plus robustes. Seuls les chronologies les plus empruntées seront conservées. En bref cette nouvelle méthode peut être résumée comme suit :

- Récupération de données et répartition du corpus par périodes. Tout comme pour la méthodologie générale, le jeu de données est divisé en périodes plusieurs années,ventre 3 et 5, se chevauchant pour lisser l'évolution et détecter le plus d'événements possible.
- Extraction de termes et plongements de mots. A cette étape, un ensemble de mots-clés est déterminé à partir des publications de chaque période. Ensuite, ils sont projetés dans un espace de grande dimension, un pour chaque période, en utilisant des techniques de plongement des plus avancées de la littérature.
- Détection de domaines. Cette partie est vraiment originale car elle consiste à déterminer une structure hiérarchique des mots-clés avec possibilité de chevauchements entre les groupes de termes. Il en résulte la génération de quasi-dendrogrammes.
- Correspondance inter-temporelle. Les cartes évolutives sont également le résultat d'une nouvelle méthode. Ils sont produits à partir de requête d'utilisateurs en utilisant l'alignement d'un ensemble de quasi-dendrogrammes successifs.

Nous avons également créé des cahiers Jupyter interactifs qui permettent de générer des quasi-dendrogrammes et des cartes évolutives personnalisés. En fait, ils visent à fournir de l'interactivité pendant le processus de génération. Ils seront déposés sur un dépot git ainsi que le code qui sera open source.

L'un des principaux problèmes de cette nouvelle méthodologie de construction de carte évolutive est son évaluation. En effet, il est vraiment difficile de déterminer les propriétés générales ou les caractéristiques que toute carte évolutive devrait respecter. Comme pour les autres méthodes, la partie évaluation est laissée à l'expert et la qualité des cartes est basée sur des propriétés de construction propres à la méthode. Ainsi, un travail futur qui devrait être fait est une comparaison appropriée avec d'autres types de carte évolutive. Cela signifie déterminer une méthodologie standard dans lequel toutes les méthodes pourraient s'intégrer et proposer des évaluations pertinentes basées sur cette méthodologie. C'est délicat car toutes les cartes n'ont pas le même but et ne cherchent pas à montrer la même chose.

Aussi pour améliorer la méthode proposée qui utilise de nombreux outils à différentes étapes, tous autant importants, une étude approfondie peut être menée pour bien identifier les impacts des choix effectués pour construire les cartes évolutives et trouver le meilleur

réglage possible. Et bien sûr, cela devrait être mené avec l'aide direct d'historiens et de philosophes des sciences. A cet effet, une interaction avec l'utilisateur apparaît également nécessaire pour analyser les structures produites et des outils de navigation interactifs semblent indispensables à court terme. En particulier il serait bien de pouvoir zoomer et dézoomer sur la structure de la science en se déplaçant plus haut ou plus bas dans les hiérarchies.

Acknowledgement

Je voudrais tout d'abord remercier mes deux encadrants Zoltan Miklos et David Gross-Amblard pour leur implication tout au long de cette thèse et notamment pour leurs précieux conseils de relecture.

Je remercie également Pierre Gançarski et Christine Largeron pour avoir accepté d'être les rapporteurs de ma thèse et d'avoir pris de leur temps pour apporter des remarques détaillées et constructives sur ces travaux.

Je tiens à remercier David Chavalarias et Elisa Fromont pour avoir fait parti de mon comité de suivi et avoir accepté de participer à mon jury de thèse.

Merci aussi à toutes les personnes avec qui j'ai pu travailler et échanger pendant ces trois ans, notamment mes collègues de l'équipe Druid.

À mon grand père Alain.

Table of Contents

INTRODUCTION

Context and motivation

Epistemologists would like to understand the evolution of science and knowledge in general. If an expert relatively knows a bit of history and evolution of its own specific field, it becomes impossible to have an historical knowledge of wide scientific domains. Moreover, now with the hyper-specialisation of the scientific domains and the multiplication of the theories and axes of research, no one can pretend to be able to follow the evolutionary mechanisms that are at stake. This requires an automatic method to extract and analyse leading theories of scientific fields.

Historically there have been three major models of evolution and change between one or more scientific theories.

The first major model, associated to Popper's view, considers that scientific progress is achieved through a falsification of incorrect theories and the adoption instead of theories which are progressively closer to truth. Science evolves in a linear fashion by accumulation of facts. The physics of Aristotle was subsumed by the classical mechanics of Newton subsumed itself by the relativity of Einstein, etc.

The main challenging model, proposed by Kuhn, sees science not as a linear progress but separated in two-phase cycles, "normal" science and "revolutionary" science. If the first phase is a calm period evolving similarly to Popper's view, the second, in which new theories based on new assumptions are tested, marks the separation between paradigms. Paradigms are dominant structures of thought and practices and represent entirely different and incommensurate (non comparable) assumptions about the universe. Only little remains from the previous paradigm when a shift occurs.

Another extreme model, promoted by Feyerabend, refutes the existence of consistent methodologies used by all scientists at all times and argued harshly against the notion that falsification was ever truly followed in the history of science. He defended his position by noticing that in practice scientists arbitrarily consider theories to be accurate even if they failed many sets of tests. The same appears for some forms of knowledge that are

1. https://en.wikipedia.org/w/index.php?curid=1677438

considered in turn as science and non-science.

Many other theories have been proposed to nuance these tree main models. They come with different variations in the representation of scientific change, the notion of progress and the relation between theory and truth.

No matter the driving forces behind science evolution, the scientific method that produces science nowadays requires to confront theories to the reality by empirical validation. These results are presented to the community through peer reviewed publications. Hence analysing the scientific papers produced by a domain appears to be good way to determine the theories involved and studying the evolution of these publications could give an access to the processes of evolutions.

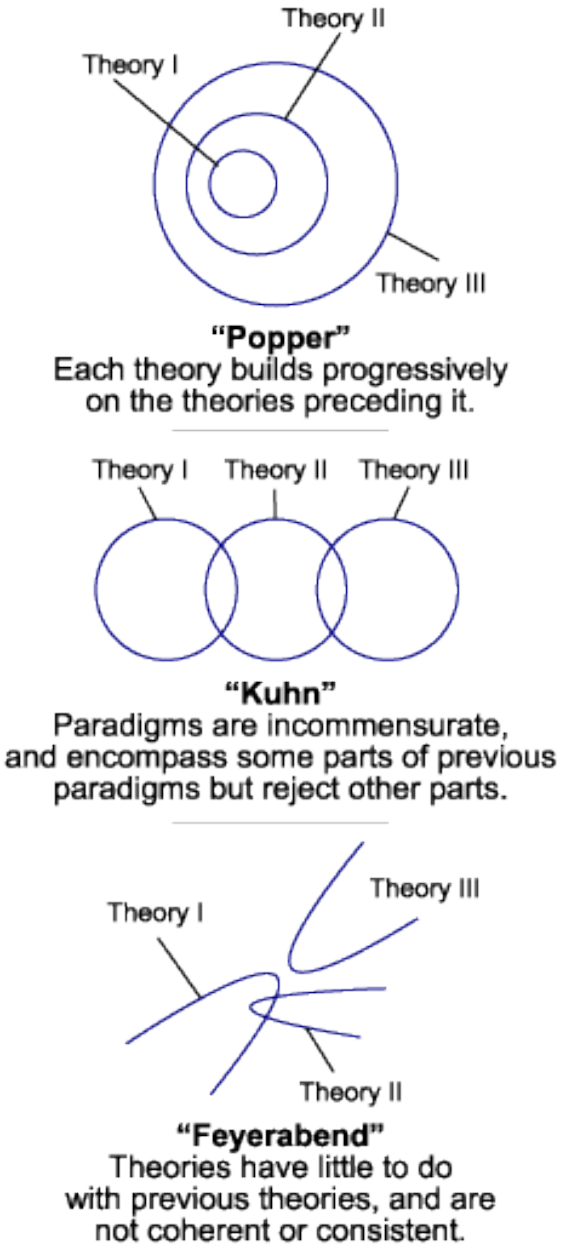


Figure 1: Three models of change in scientific theories, depicted graphically to reflect roughly the different views associated with Karl Popper, Thomas Kuhn, and Paul Feyerabend. CC BY-SA 3.0 [1]

Main challenges and problem presentation

Pioneers in quantitative epistemology tried to provide automatic tools to map the scientific landscape to allow historians of science to fully understand the evolution of science. Firsts tried to build maps based on citation index when more recent works got down to the study of the text itself which allowed a thinner granularity in the maps produced. The recent promotion for open science lead to the availability of massive corpora of scientific production which requires state of the art data-mining methods to

extract patterns from these large databases such as Wiley, dblp or Web of Science that gather millions of scientific publications.

To automatically reconstruct evolutionary structures, scientists need to combine several tools of Natural Language Processing and Text Mining in a workflow that can be divided in several steps and that is generally defined as follow:

— Data Retrieval and corpus splitting by periods. Typical periods to study recent evolution of science are several years, between 3 and 5, with a sliding window over the periods to have an overlap on the data. As science evolves continuously, splitting the data by periods is arbitrary and having a split in the middle of a particular event could make it undetected in the phylomemy, the overlap is here to prevent such behaviour.

— Term extraction and similarity matrix. This step transforms the set of documents of each period into a weighted term matrix. This matrix aims to represent the relationship between key terms, typically n-grams. The classical representation of this relationship is based on the co-occurrence of the words in a document.

— Topic detection. It consists in detecting sets of strongly semantically related terms within term graphs, the graphical counterpart of the term matrices.

— Inter-temporal matching. It matches topics from different periods, usually by using Jaccard index, to generate alignments representing the temporal evolution of the topics. This will lead to splits in the structures, which are common in a phylogenetic structure, but also to merges and other typical events specific to this type of evolutionary map. Usually an interaction with the map allows experts to customize the workflow by changing data (for example removing or adding a term in a topic) and parameters (for example the time interval for dividing the document collection).

This is the main challenge of the ANR EPIQUE project, in which this thesis fits. Project partners have been pioneers in the reconstruction of science dynamics mining corpora at large scale (Chavalarias and Cointet 2013) and they have shown that we can characterize quantitatively the different phases of the evolution of scientific fields and automatically build "phylomemetic" topic lattices representing this evolution (Figure 2). Phylomemetic lattice comes as an analogy with the well known phylogenetic tree of natural species.

The project also involve philosophers of science, who perceive phylomemetic lattice structures as a tool for testing general accounts of progress and change in science. They

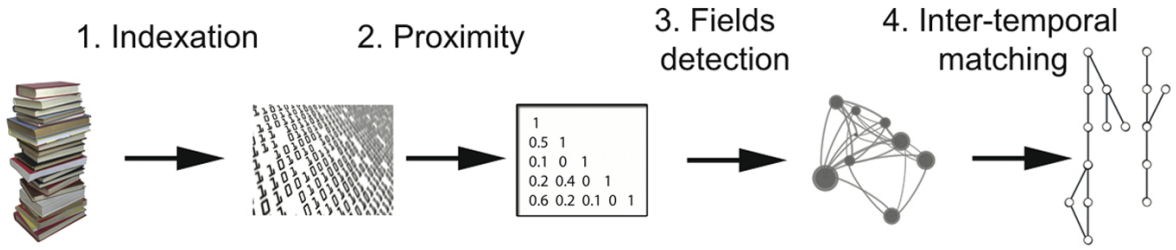


Figure 2: An existing workflow of phylomemy reconstruction (Chavalarias and Cointet 2013).

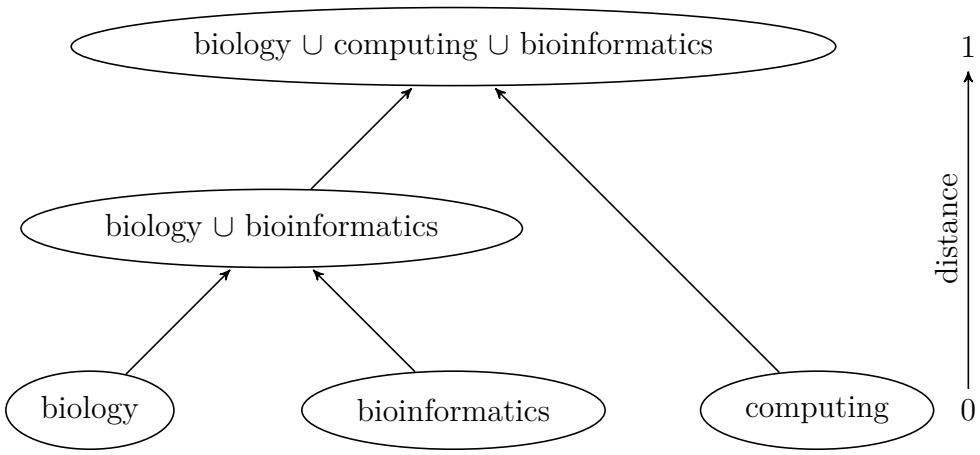
will play a role in validating the lattices produced in the context of the project. They especially underlined the necessity to map not only the evolution of scientific fields but also the global structure of science.

Contributions

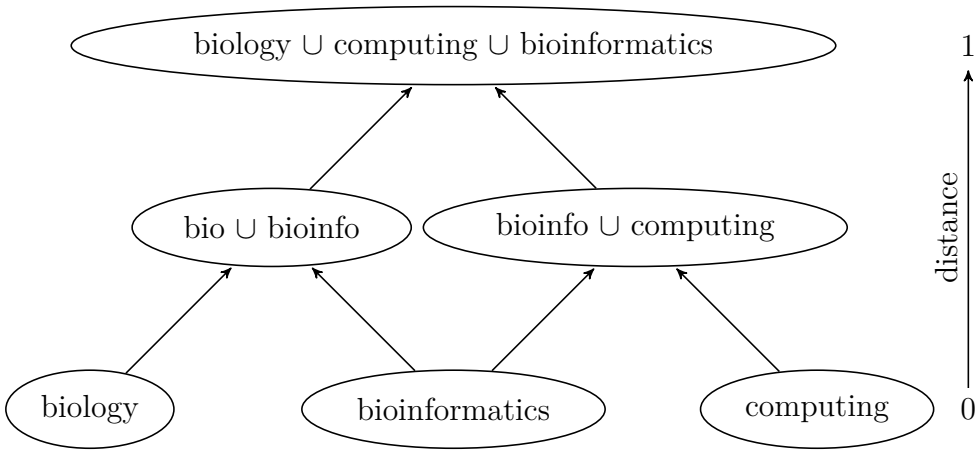
In this thesis we propose an alternative workflow of generating evolutionary maps of scientific domains using state of the art techniques. It will allow to have different views on the structure of Science which constitutes interesting new tools for philosophers and historians of Science.

In the framework of analysing only the raw text of the scientific papers and not the citation index, a possible approach is to construct a word embedding of the key terms in publications, that is, to map terms into a high-dimensional space such that terms frequently used in the same context appear close together in this space (Section 1.1.3). Identifying for example the denser regions in this space directly leads to insights on the key terms of Science. Moreover, building a dendrogram of key terms using an agglomerative method is typically used (Chavalarias and Cointet 2013; Dias et al. 2018) to organize terms into hierarchies (Section 1.2). This dendrogram (Figure 3a) eases data exploration and is understandable even for non-specialists of data science.

Despite its usefulness, the dendrogram structure might be limiting. Indeed, any embedding of key terms has a limited precision, and key terms proximity is a debatable question. For example, in Figure 3a, we can see that the *bioinformatics* key term is almost equally attracted by *biology* and *computing*, meaning that these terms appear frequently together, but in different contexts (e.g. different scientific conferences). Unfortunately, with classical agglomerative clustering, a merging decision has to be made, even if the advantage of one



(a) A classical dendrogram, hiding the early relationship between *bioinformatics* and *computing*.



(b) A **quasi-dendrogram**, preserving the relationships of *bioinformatics*.

Figure 3: Dendrogram and quasi-dendrogram for the structure of Science.

cluster on another is very small. Let us suppose that arbitrarily, *biology* and *bioinformatics* are merged. This may suggest to our analyst (not expert in computer science) that *bioinformatics* is part of *biology*, and its link to *computing* may only appear at the root of the dendrogram. Clearly, an interesting part of information is lost in this process.

The first contribution of this thesis is the proposition of a new **Overlapping Hierarchical Clustering algorithm** (Chapter 2) that combines the advantages of hierarchies while avoiding early cluster merge. Going back to the previous example, we would like to provide two different clusters showing that *bioinformatics* is close both to *biology* and *computing*. At a larger level of granularity, these clusters will still collapse, showing that these terms belong to a broader community. This way, we deviate from the strict notion of trees, and produce a directed acyclic graph that we call a quasi-dendrogram (Figure 3b).

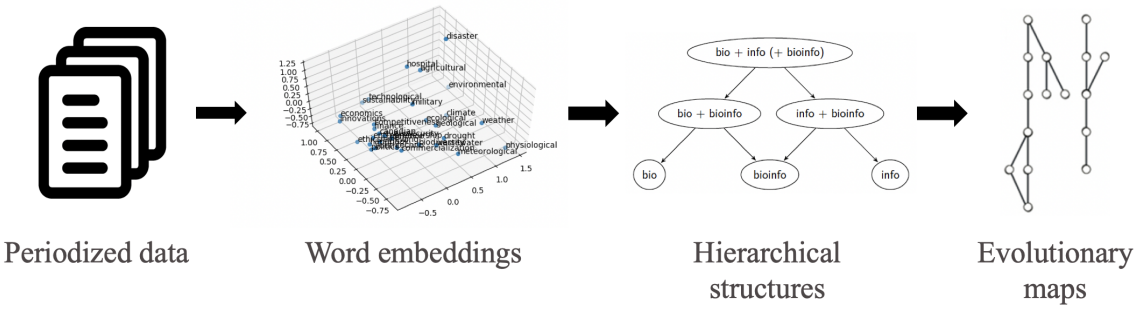


Figure 4: New workflow of evolutionary maps reconstruction.

For this purpose we define a density-based merging condition to identify these clusters.

The second contribution is the definition of a new **similarity measure** (Chapter 3) to compare our method with other, quasi-dendrogram or tree-based ones. With this similarity measure and its variants we show through extensive experiments on real and synthetic data that we obtain high quality results with respect to classical hierarchical clustering, with reasonable time and space complexity.

Finally to complete the new workflow (Figure 4) the third contribution is the generation of **evolutionary maps** (Chapter 4) as the result of the inter temporal study of the quasi-dendrograms. We produce these evolutionary maps based on queries made from an expert to interact with the flow of evolution of topics of science determined by quasi-dendrogram alignments.

The results presented in this thesis have been published in the following venues:

— Ian Jeantet (2018), « Study science evolution by using word embedding to build phylomemetic structures », *in: BDA - Conférence sur la Gestion de Données – Principes, Technologies et Applications*. [PhD paper]

— Bernd Amann, David Chavalarias, Ian Jeantet, Thibault Racovski (2019), « Digital history and philosophy of science: The reconstruction of scientific phylomemies as a tool for the study of the life sciences », *in: International Society for the History, Philosophy and Social Studies of Biology meeting* [Workshop]

— Ian Jeantet, Zoltán Miklós, and David Gross-Amblard (2020), « Overlapping Hierarchical Clustering (OHC) », *in: International Symposium on Intelligent Data Analysis*, Springer, pp. 261–273. [Regular paper]

— Ian Jeantet, Thanh Trung Huynhn, Zoltán Miklós, David Gross-Amblard and Hung Nguyen Quoc Viet, « Mapping the evolutionary history of scientific concepts ». [To be submitted]

Chapter 1

State of the art

This state or the art Chapter will present a wide range of possible methods that can be used at different steps of the workflow. Even if the contributions are not at this part, we need to have an overview or the techniques of Natural Language Processing that allow to extract and represent information from a text, basically key terms and their relationships. Then we will see several techniques of clustering and their field of application that could determine topics among the keyterms. We will challenge these clustering techniques with our overlapping hierarchical clustering algorithm detailed later in Chapter 2. Also we will present classical evaluation techniques to see if they can or cannot be applied to our OHC algorithm. This will be explored in Chapter 3. Finally we will look at more general methods that already produce evolutionary maps to see on how the point of views and insights they allow are different from the one we propose in Chapter 4.

1.1 Text analysis

This first Section will explore state of the art methods that one can use in the two first steps of the evolutionary map reconstruction that is to determine key terms in a text, scientific publication in our scenario, then how to determine the relationship between terms. Also before that we need to explore the methods that need to be used to make a text readable to a machine.

1.1.1 Text preprocessing

Following the good practices given by Dias et al. 2018, in this section we discuss the possibilities of text reprocessing to transform text in human language from scientific papers to a machine readable format.

Language selection

First it seems logic that one need to select articles written in only one language. Nowadays as most of the them are written in English it is the obvious language to select to study recent datasets. It is also the most studied language hence the tools represented later will be more efficient on English terms. However for a specific study, especially on older datasets, another language could be considered.

Text selection

For most of available datasets one still don't have access to the real content of the articles for copyright reason but to the title and optionally their abstract. For need as much text as possible so the title and abstract are concatenated for each article and considered as a unique entry.

From the raw text as a unique string we need to identify the words it contains and give them the correct form. It can be done in several ways.

Text cleaning

Apart from the words, a text contains many other characters that give a specific format to the text depending on the grammar of the considered language.

If it greatly eases the understanding of a text by a human reader, it only complexifies the work of an algorithm. Hence the following methods can be applied to uniform the words of a text. First one can simply convert all letters to lower (or upper) case. Then the numbers can be converted into words or be removed if they are not relevant for the study. Also contractions such as *it's* can be replaced by their non-contracted form, *it is* in our example. A list based on Wikipedia's List of English contractions [1] is provided by Dias et al. 2018. Finally punctuation and whitespaces, hidden characters such as tabulations '\t' or line breaks '\n' that represent a horizontal or vertical space and aerate the text, can optionally be removed. One must be careful that punctuation is essential for the comprehension of a text and cannot be removed for certain tasks.

1. `https://en.wikipedia.org/wiki/Wikipedia:List_of_English_contractions`

Tokenization

One of the key steps is tokenization. Tokenization is the process of splitting a sequence of characters into a sequence of small pieces called tokens. Words and numbers, i.e. alphanumeric chunks, are typical tokens but punctuation and white spaces can be consider as such and be included in the resulting list of tokens. Tokens can also mix letters and special characters such as hyphen depending on the rules that are applied for the tokenization. The result of the tokenization depends on the cleaning done on the text and will differ if the punctuation was removed prior the tokenization for instance.

Having a list of tokens is enough to be use by a machine for further processing but depending on the task to perform, one needs to use more advance techniques that can help to improve the outcomes.

Token canonicalization

Language is composed of words that can take many inflected forms. For tasks like sorting documents into different categories, it is interesting to reduce them to their root or stem form. Indeed in this case the subtleties that exist between the word variants are not important.

Stemming In linguistic stemming is the process of reducing a word to its stem form. For example the words *plants*, *planting* and *plantation* should be reduce to the root form *plant*. It is important to notice that a stem is not necessary a valid word as it is often the result of the pruning of known suffixes and prefixes. For instance the word *natural* can be reduced to the stem *natur*.

Stemming algorithms have been studied since the 1960s and the main two stemmers are the Porter stemming algorithm extended to the Snowball stemmer family (Porter 2001) and the Lancaster stemming algorithm also known as the Paice-Husk Stemmer (Paice 1990). The second one is recognized to be a 'stronger' stemmer, that is pruning more harshly the words, with its standard tuning, i.e. not changing any parameter.

Lemmatization In linguistic lemmatization is the process of identifying the canonical form (also called lemma) of a inflected word based on its meaning. Unlike stemming, lemmatization needs the context of a word to correctly identify the meaning of this word. The context is based on determining the part of speech of the word but can be extended

up to surrounding sentences and even the entire document. Common parts of speech of Indo-European languages are noun, verb, adjective, etc.

As examples:

— The word *good* is the lemma of the word *better*. This transformation is missed by a stemming algorithm.

— The word *plant* is the lemma of the word *planted* which is also identified by stemming.

— The word *meeting* can be either used as a noun of a verb and depending on the case the lemma is different, *meeting* for the noun and *meet* for the verb. While a stemmer will always select *meet*, lemmatization will try to identify the correct form based on the context.

Due to its complexity, finding an efficient lemmatization algorithm is still an open research area. Early works were mostly based on statistical models using an hand-build set of possible word-lemma pairs (Hakkani-Tür, Oflazer, and Tür 2002) when recent works are based on log-linear models (Müller et al. 2015) or neural networks (Bergmanis and Goldwater 2018; Bojanowski et al. 2017).

Then if stemmers are usually easier to implement and faster because based on rules they could be seen as less 'accurate' than lemmatization algorithms. In fact in information retrieval tasks they reduce the precision, i.e. the capacity to detect only relevant information, and increase the recall, i.e. the capacity to detect all the relevant information (Christopher D. Manning, Raghavan, and Schütze 2009).

Also for other tasks such as finding the relationship between words for syntactic or semantic reason it is not possible to perform stemming or lemmatization because we need to preserve the relationship between every inflected word forms, between *good*, *better* and *best* for instance.

Token cleaning

Finally, non interesting tokens can be removed from the text. Usually one remove single character tokens that often come from a bad detection during the tokenization process and that are non word tokens.

At this step it is also possible to remove numbers and/or stop word tokens. Stop words are extremely common words that appear to have little value in tasks like information retrieval or search engine query and hence that can be discarded to decrease the size of

the dataset and improve the performances and accuracy. Indeed, a keyword search based on words like *the* or *from* is not really useful. However the trend is to include them as much as possible and for other tasks like phrase search or machine translation they are at upmost importance and should be preserved. For example, *this movie is not good* has a different and more precise meaning than the isolated words *movie* and *good*.

There is no universal stop word list. To be not significant, a word needs to be evenly distributed in a collection of texts. Indeed if it appears with the same frequency in all the texts it cannot be used to discriminate and select specific ones in the collection. If all the texts are in the same language these words are usually function words like *the*, *at*, *which*, etc for English and combination of existing lists are often used. However for a corpus of texts on a common theme, specific words can be uniformly distributed and then should be also considered as stop words and on the contrary some function words appear rarely in texts. Another practical approach (Christopher D. Manning, Raghavan, and Schütze 2009) consists in ordering the words by collection frequency, i.e. considering the total number of occurrence of each word in all the texts, and taking the list of most frequent words as stop words.

1.1.2 Keyword extraction

Keywords are important terms and phrases within documents. They play an important role in many applications in Text Mining, Information Retrieval and Natural Language Processing and also in our workflow. With the huge quantity of documents now available it is illusory to think of being able to read them all and find information in them. For this reason it is mandatory to have an automatic approach. In keyword assignment, the keywords are chosen from a predetermined set of terms that match the most the documents. In this section we will see methods of keyword extraction that is to extract the keywords directly from the documents, as it corresponds to the task we will need to perform on scientific publications, and we will discuss the great diversity of these approaches (Figure 1.1).

Statistical approaches

Methods using statistical rules are among the simplest to detect keywords or key phrases as they don't require prior knowledge either on the domain or the language. However they are based on rules and hence can miss specific relevant keywords.

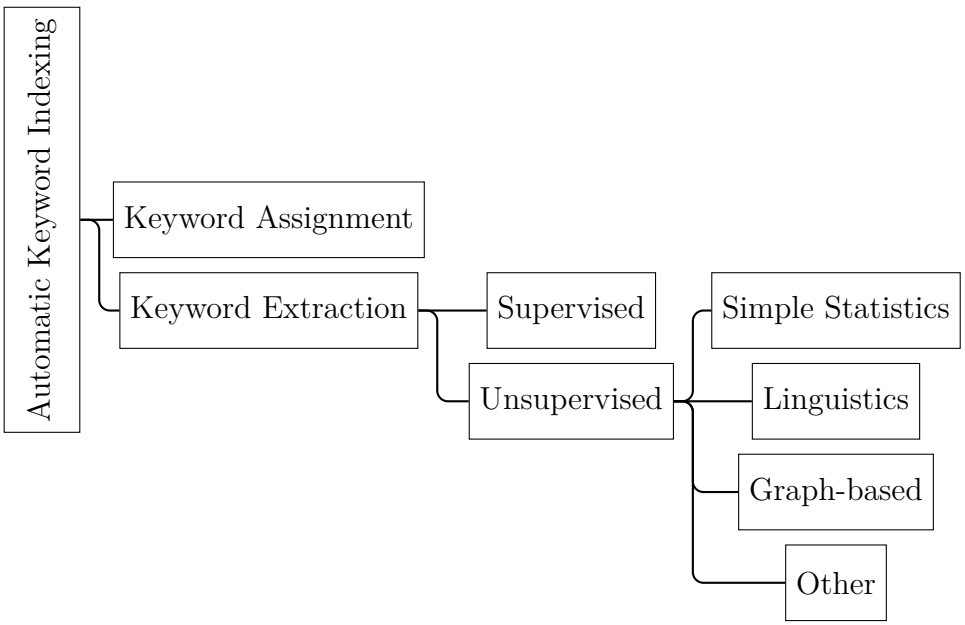


Figure 1.1: Classification of keywords extraction methods (adapted from Beliga, Meštrović, and Martinčić-Ipšić 2015).

Word frequency Sorting words by frequency allows to detect words that appear commonly in a text or corpus. It is useful to determine the main theme of a document or the most common issues from a Q&A section for instance.

However this approach consider text as a bag-of-words (Definition 1) putting completely aside the meaning of the words and the structure of the text.

Definition 1 (Bag-of-words model, Harris 1954)**.** *The bag-of-words model is used in NLP and IR as a simplification of text representation. In this model a text is represented as the multiset (the bag) of its words that is by keeping only the words and their multiplicity, excluding grammar like punctuation and even the word order.*

Hence this method will also detect frequent language-specific function words and miss synonyms, dismissing valuable information.

N-grams and word co-occurrences To understand better the semantic structure of a text, one can use statistics to detect words that appear frequently together. In this case they can be considered as a unique entity which allows to extract more complex terms from the documents.

It can be frequent adjacent words in the case of word n-grams (Definition 2). The sequence *new york times* can be composed of 3 separated words but one can also consider

the bi-gram *new york* which refers to the city, keeping *times* aside, or the tri-gram *new york times* which refers to the journal. In each of these cases, one can see that the meaning of the words radically changes which implies a different understanding of the text.

Definition 2 (n-gram)**.** *In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application.* [2]

In a corpus, statistics can also be used to detect words that frequently co-occur in the same texts without having to be adjacent. If the relation between the words is less direct, they still have a semantic proximity.

TF-IDF TF-IDF, that stands for *term frequency–inverse document frequency*, is a family of statistics that aims to measure how important a word is to a document in a corpus. TF-IDF is the product of two statistics, the *term frequency* and the *inverse document frequency*. It exists various ways to determine these values.

The *term frequency* is the same measure as explained in the previous paragraphs. It counts the number of occurrence of a term in a document. If one wants to identify the documents that correspond the most to a query like *the brown fox*, relevant documents should be the documents where these three words have the most weight. Luhn 1957 first stated that the weight of a term that occurs in a document is proportional to the term frequency. If the raw count is the simplest choice, adjustments are often made to take the length of the documents into account for instance. In this case one can divide the raw frequency of a term t in a document d by the number of words in the documents (Equation 1.1).

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1.1}$$

However for common terms such as *the* in our example the term frequency will tend to highlight documents that frequently contain this term without distinction with the two other terms that are more specific. To overcome this issue and give more weight to the terms that occur rarely, Jones 1972 introduced a term-specificity statistic called *inverse document frequency* on the idea that the specificity can be quantify as an inverse function of the number of documents in which it occurs. Classical *inverse document frequency* can be obtained by taking the logarithm of the inverse fraction of the number of documents

2. `https://en.wikipedia.org/wiki/N-gram`

containing the term t by the total number of documents $N = |D|$ of the corpus D (Equation 1.2). Similarly to the *term frequency*, it exists many variants for the *inverse document frequency*.

$$idf(t, D) = log \frac{|D|}{|\{d \in D | t \in d\}|} \tag{1.2}$$

As the product of the two measures (Equation 1.3), a high TF-IDF is reached when a term appears frequently in a document and, on the contrary, if this term appears in many documents, the ratio in the logarithmic function will be closer to 1 and hence the TF-IDF closer to 0.

$$tf\text{–}idf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{1.3}$$

Extracting keyterms based on this measure is interesting, they have a special meaning with regard to the corpus. In particular they are frequent in the documents but certainly not evenly distributed as they need to appear in less documents as possible. As an example, one of the state-of-the-art TF-IDF functions is the ranking function called Okapi BM25 (Robertson and Zaragoza 2009; Whissell and Clarke 2011).

Linguistic approaches

To improve statistical approaches one can use linguistic information about the texts and the words. In combination with a method seen above, one can use it to reduce a preselected set of keyterms.

This linguistic information can be the part-of-speech (PoS) of the words. One can give a higher score to nouns as they often carry more information than other type of words.

Also the position of a word in a text can give some information about its importance (X. Hu and Wu 2006). If a word appears in the first and last paragraphs of a document, usually the introduction and the conclusion, it is generally more important. Same for words of the leading and concluding sentences of a paragraph. Detecting conjunctions and discourse markers that connect expressions and sentences can also be used for this purpose. X. Hu and Wu 2006 stated that their method called Position Weight Inverse Position Weight (PWIPW) outperforms standard TF-IDF.

Graph-based approaches

A graph is defined as a set of vertices with some connections between them, the edges. A graph is a well established mathematical structure that facilitates an efficient exploration of the structure and the relationships between the vertices.

Graph generation They are many ways to represent a text as a graph. Usually the words are the vertices and the edges represent the relationship between the words but depending on the goal, different methods can use different linguistic units as vertices and the generated graph can be directed or undirected, weighted or unweighted. Beliga, Meštrović, and Martinčić-Ipšić 2015 and Firoozeh et al. 2020 gave an overview of the different principles on which the edges between the vertices of the graph can be established.

— *Co-occurrence graphs.* They connect words co-occurring in a window of a given number of words in the text. This type of graphs are used in particular by TextRank-like algorithms (Mihalcea and Tarau 2004). They can also connect all the words of a sentence, paragraph, section or even document that hence will appear as a clique (Definition 11) in the graph.

— *Syntax dependency graphs.* They connect words according to the syntactic relationship of the part-of-speech they belong (Huang et al. 2006; Liu and F. Hu 2008).

— *Semantic-based graphs.* They connect words that have similarities in their meaning, spelling, that are synonyms, antonyms, homonyms, etc. For instance Grineva, Grinev, and Lizorkin 2009 exploit semantic relations of terms extracted from Wikipedia to built such graphs.

Graph analysis From such a graph, keyword extraction works by measuring how important is a vertex to the graph, i.e. a word to the text, based on information given by the structure of the graph. Beliga, Meštrović, and Martinčić-Ipšić 2015 and Firoozeh et al. 2020 also presented several methods of the literature that serve this purpose. Unsupervised methods to get a list of keywords are divided into two main categories: clustering and ranking algorithms.

— *Clustering algorithms.* They work by grouping together nodes of the graph that are semantically related. For tasks such as indexing, Ohsawa, Benson, and Yachida 1998 showed that, with their KeyGraph algorithm, a clustering based on a co-occurrence graph outperforms TD-IDF and N-grams approaches. They also proved KeyGraph to be a content-sensitive and domain-independent indexing device. An-

other approach from Grineva, Grinev, and Lizorkin 2009 uses community detection techniques to identify relevant words and discard irrelevant ones.

— *Ranking algorithms.* This approach is the most common and aims at giving a rank to the vertices of the graph based on the global information represented by entire graph. The technique was introduced by Mihalcea and Tarau 2004 with their TextRank algorithm. They adapted the PageRank algorithm on a weighted co-occurrence graph and the result appeared as a strong baseline for this type of algorithms. It exists many other variants and an extensive presentation of them is given by Beliga, Meštrović, and Martinčić-Ipšić 2015.

Supervised approaches

One can see keyword extraction as a classification problem that aims to labelled each keyword candidates as either keyword or non-keyword. One of the first classifiers is the Keyphrase Extraction Algorithm (KEA) proposed by Frank et al. 1999. It uses a Naïve Bayes algorithm based on statistical and informational features for the training and it had many improvement over the years. It exists many other supervised methods that are presented in details by Firoozeh et al. 2020. One can notice the work of Krapivin et al. 2010 that uses NLP techniques to improve Machine Learning approaches for the automatic keyword extraction from scientific papers as it is closely related to the subject of this thesis. They showed that this outperforms the state-of-the-art KEA without the use of controlled vocabularies.

In general these supervised methods compete with unsupervised ones but the training data requirement is a strong limitation as it has to be generated manually which is very costly. Sterckx et al. 2016 pointed out the difficulty to find reliable training data which has consequences on the outcomes.

Discussion on the methods

According to Firoozeh et al. 2020, due to its difficulty Keyword Extraction is still an open area of research and there is no single efficient method that would work in all applications. Each one has its pros and cons with a *F-measure* around 50% for the highest reported scores in the literature. The *F-measure* is an accuracy measure that consider both the *precision* and the *recall* to compute its score. The *precision* is the capacity of a method to retrieve only relevant information, in our case relevant keywords, and the

recall is the capacity of a method to retrieve all relevant information. More details about the *F-measure* are given in Section 1.2.5.

If a supervised approach can be a good choice when annotated data are available, for the vast majority when it's not the case, unsupervised approaches are a better choice. Among them TF-IDF has been widely used and recent approaches combine it with other methods such as graph-based ones. Also TF-IDF can be comparable to such graph-based methods on specific cases. In fact TF-IDF tends to identify specific and representative words when graph-based methods also search for completeness.

It is also possible to remove irrelevant keywords afterwards using termhood processing. Similarly to the word frequency method the idea is to remove evenly distributed keyword candidates. It doesn't matter if they are technical words or phrases, if they appear in all the documents they are considered neutral with respect to the given corpus (Chavalarias and Cointet 2013; Van Eck and Waltman 2011). If the corpus is related to a specific domain one can also asks to expert to finish the cleaning of irrelevant keywords (Chavalarias and Cointet 2013).

1.1.3 Word embedding

We explored methods useful to format a text and to extract keyterms from a corpus. Now we need a method that we can use to determine the meaning of the these terms and the relation it exists between them. It will be necessary for the second step of our workflow. Word embedding gives a standard representation of words or terms which serves this purpose. It is a general technique in Natural Language Processing (NLP) where words or phrases from a vocabulary are mapped to a high-dimensional vector space. It aims to quantify and categorize semantic similarities between language terms based on their distributional properties in large samples of data. For instance the word "kangaroo" will be more related to the word "Australia" than to other words such as "Russia" or "watermelon". We will see in this section a review of different possible methods of word embedding.

Co-occurrence based approaches

Traditionally in the literature a term is represented by the "bag of documents" in which the term occurs. This representation is the Document Occurrence Representation (DOR) where a term vector contains the document weights that correspond to the contribution of each document of the corpus to the semantic specification of the term. Latent Semantic

Analysis (Dumais 2004) uses this representation. The key steps of LSA are computing this term-document matrix, using typically TD-IDF as weights, and performing a dimension reduction using Singular Value Decomposition (Golub and Reinsch 1971) on this $|T| \times |D|$ matrix. The resulting $k \times k$ matrix is the best k-dimensional approximation of the original matrix using the least squares method. Now each term (and document) has a k-dimensional vector representation.

As shown by Lavelli, Sebastiani, and Zanoli 2004, basic DOR is outperformed by a second approach, the Term Co-Occurrence Representation (TCOR) which differs from DOR by representing the terms as "bags of terms", i.e. vectors of term weights instead of document weights. As in DOR these weights correspond to the contribution of every term in the vocabulary to the semantic specification of the represented term. The classical $|T| \times |T|$ co-occurrence matrix is a typical example of the TCOR representation. The Hyperspace Analogue to Language (Lund and Burgess 1996) model also uses this term-term matrix representation. The cells of the matrix are updated according to the co-occurrence of terms in a 10 word window sliding over the text. The cell corresponding to the co-occurrence of two terms is increased by $\Delta = 11 - d$ where d is the distance between the two terms in the window. One of the strength of the TCOR is to be "corpus independent" in the sense that the size of the matrix does not depend on the number of documents in the corpus.

Neural Network Language Models (NNLM)

The main power of neural network based language models is their simplicity. The same model with only few changes can be used for projection of many type of signals including language. Also these models performs implicitly clustering of word in low-dimensional space which gives us a robust and compact representation of words.

Among the first neural models used in NLP, Bengio, Ducharme, et al. 2003 proposed a probabilistic feedforward NNLM which consists of input, projection, hidden and output layers. The n previous words are represented as one-hot vectors (Definition 3) of size V where V is the size of the vocabulary. The input layer is projected to a projection layer P of a $n \times D$ dimension that uses a shared projection matrix noted C in Figure 1.2. Then these n projected vectors feed a hidden layer of a H dimension that use the hyperbolic tangent as activation function. Finally the hidden layer is used to compute the probability distribution over all the words in the vocabulary through a softmax activation function.

It results in an output vector of size V.

$$Q = n \times D + n \times D \times H + H \times V \tag{1.4}$$

The computational complexity per training Q is as shown as in Equation 1.4 where the dominating term is $H \times V$. However, as Tomáš Mikolov, K. Chen, et al. 2013 recall us, several alternatives are existing to go down the complexity to $H \times log_2(V)$. Hence the dominating term becomes $n \times D \times H$.

Definition 3 (One-hot encoding)**.** *A one-hot vector is a* $1 \times |V|$ *matrix used to distinguish each word in a vocabulary of size* $|V|$ *from every other word. The vector consists of 0s in all cells with the exception of a single 1 in a cell used uniquely to identify the word.*

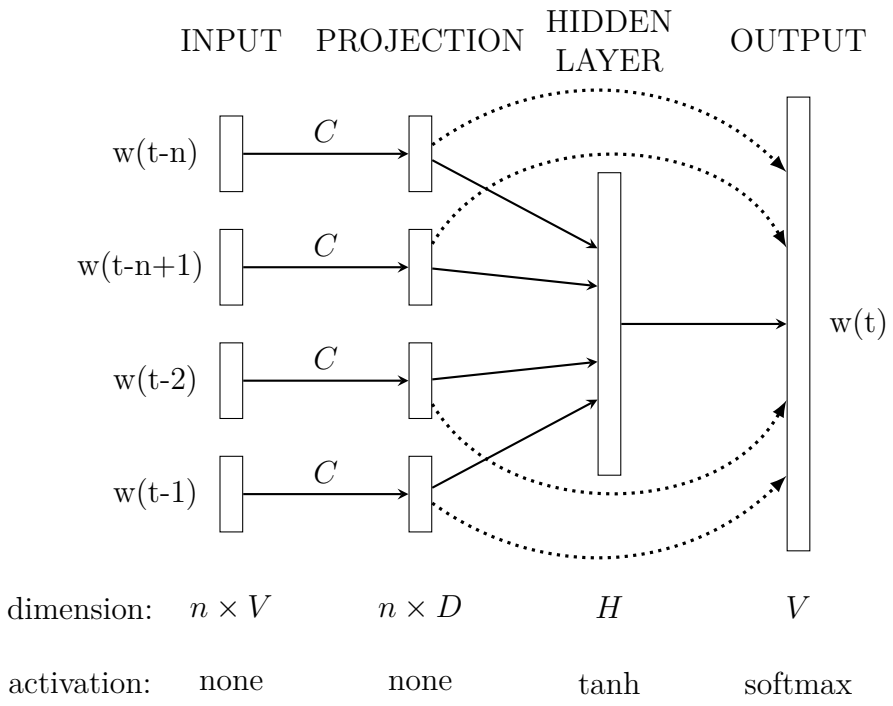


Figure 1.2: Neural architecture of the feedforward NNLM (adapted from Bengio, Ducharme, et al. 2003).

Language models using a recurrent neural networks (RNNLM) have been proposed by Bengio, LeCun, et al. 2007, Tomáš Mikolov, Karafiát, et al. 2010 and Tomáš Mikolov, Kombrink, et al. 2011. Unlike feedforward neural networks a RNN has no projection layer which means this model is independent to the specified order n of the feedforward NNLM. The architecture of the basic RNN model, also called Elman network, is shown in Figure

1.3. In RNN the model uses only the previous word as input but also a recurrent matrix (R in the figure 1.3) which contains the previous weights of the hidden layer. This recurrent matrix and the input word represented by a one-hot vector are concatenated and connected to the hidden matrix H. It simulates a time delayed connection and represents some kind of short term memory. The same well known activation functions (sigmoid and softmax) are used in this model and the network is trained by using the standard back-propagation technique. After some optimisations mainly from Tomáš Mikolov, Kombrink, et al. 2011, the computational complexity per training is

$$Q = H \times H + H \times V \tag{1.5}$$

As said previously for the feedforward NNLM, the term $H \times V$ in Equation 1.5 can be reduce to $H \times log_2(V)$ then most of the complexity comes from $H \times H$.

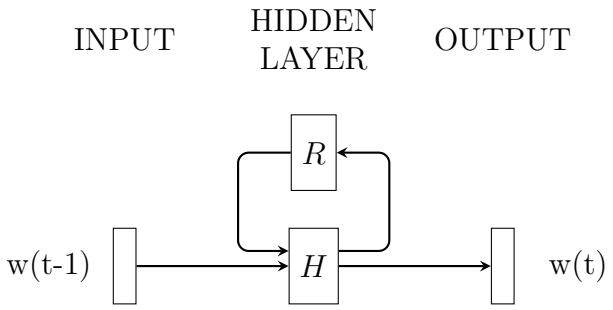
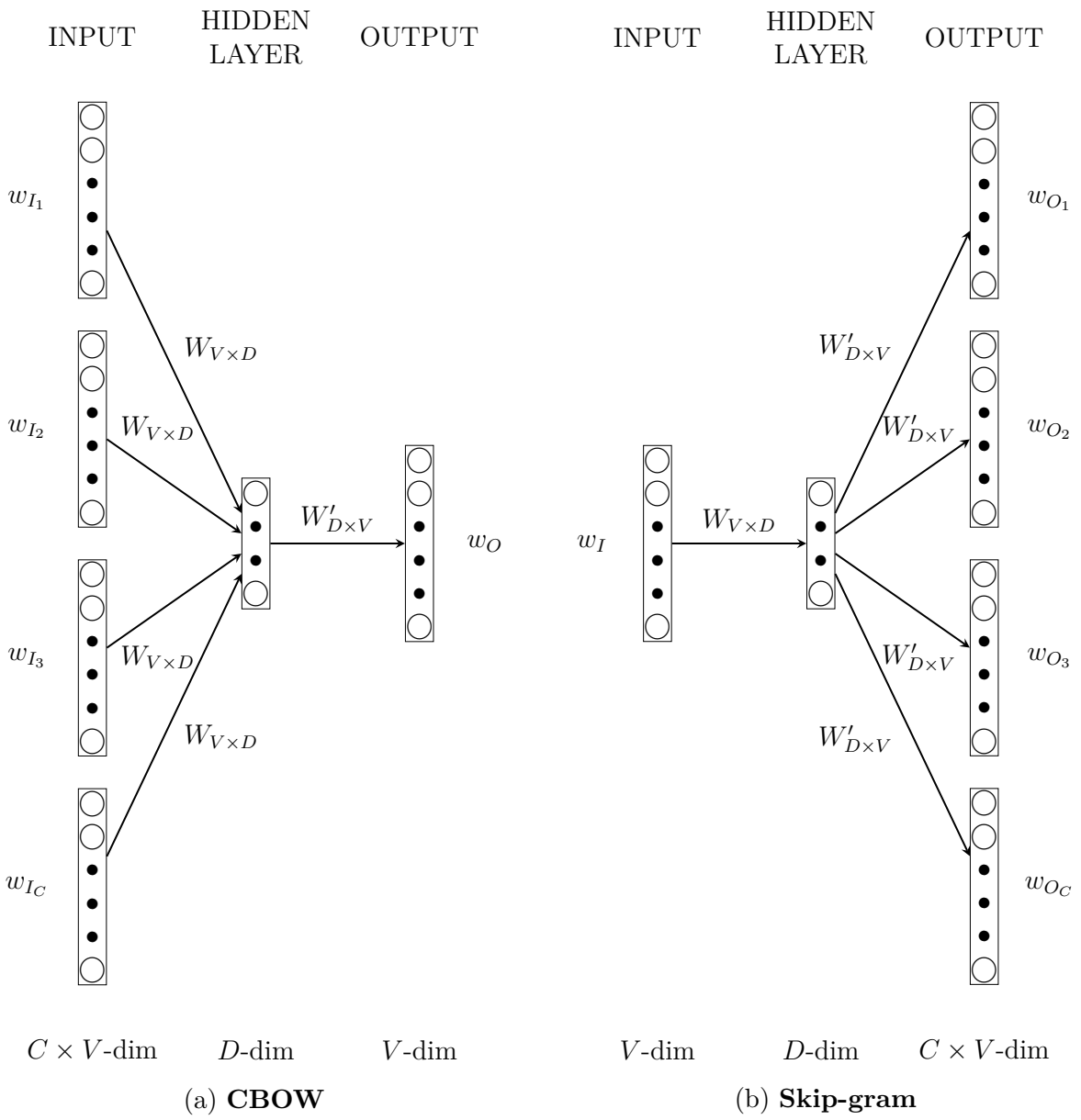


Figure 1.3: Simple recurrent neural network architecture.

Log-linear models

The main problem of the existing NNLMs is their computational complexity which is caused by a non-linear hidden layer in the models. This complexity is a real challenging problem when it comes to deal with very large amount of training data and high-dimensionality of the word vectors. To reduce this complexity, Tomáš Mikolov, K. Chen, et al. 2013 proposed to use simpler model architectures: Continuous Bag-of-Words (CBOW) and Skip-gram models.

The goal of these two models is to tell us the probability for every word in our vocabulary of being the nearby word of a given word we chose. In Figure 1.5, nearby means 2 words behind or ahead from the position of the chosen word in the sentence. More generally we note this window size C as in Figure 1.4.



(a) **CBOW** (b) **Skip-gram**

Figure 1.4: (a) The CBOW architecture predicts the current word based on the context when (b) the Skip-gram predicts surrounding words given the current word (adapted from Tomáš Mikolov, K. Chen, et al. 2013).

The word pairs as shown in Figure 1.5 will be used to train a neural network. At the third step (the third line of the figure) the words *the*, *quick*, *fox* and *jumps* will be inputs for the CBOW model when the expected output *brown* will be used for backpropagation learning. As the Skip-gram model works in mirror of the CBOW model, *brown* will be the input and the 4 other words the expected outputs. It is important to notice that

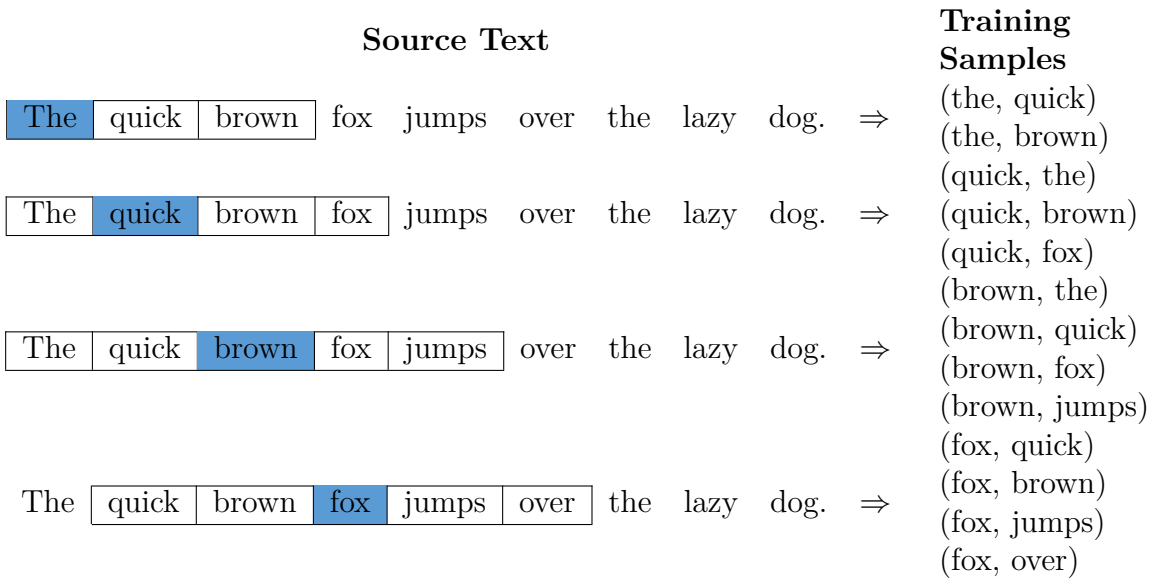


Figure 1.5: Example of some of the word pairs taken from the sentence *The quick brown fox jumps over the lazy dog* (adapted from McCormick 2016).

under the bag-of-words assumption, the order of the context words has no influence on the prediction.

Let see in details how the CBOW model works by looking at the forward propagation (how the output is computed from the inputs) and the backpropagation (how the weight matrices are trained).

For the forward propagation one assume that the input and output weight matrices are known, respectively noted W and W' in Figure 1.4. The output of the hidden layer h is the average of the sum of the inputs vectors weighted by the matrix W which is formally expressed by

$$h = \frac{1}{C} W \cdot \sum_{i=1}^{C} w_{I_i} \tag{1.6}$$

In fact the average vector of the inputs is projected on a space of dimension D. One can notice that in this model there is no activation function for the hidden layer. From the hidden layer to the output layer, using each column of the weight matrix W' one can compute a score for each word of the vocabulary by calculating

$$u_j = {W'_{*,j}}^T \cdot h \tag{1.7}$$

where $W'_{*,j}$ is the j-th column of W'. u_j is a measure of the match between the context and the candidate target word. Finally the output w_O is obtained from the output layer by passing u_j through the softmax function to calculate its j-th component.

$$w_{Oj} = p(w_j|w_{I_1}, ..., w_{I_C}) = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{1.8}$$

Each component represents the probability of the word w_j of the vocabulary to be the output word knowing the context so the output vector is in fact a probability distribution and not a one-hot vector.

Now let see how the weights of the two matrices are learnt. As said before, the word pairs as shown in Figure 1.5 are used to feed the model. The training objective is to match the probability distribution calculated by the model w_O with the one-hot vector representing the target word w_T. To do this every distance could be used but the most common one is the cross entropy.

Definition 4 (Cross entropy)**.** *Considering p and q two discrete probability distributions, the cross entropy is defined as*

$$E(p, q) = -\sum_{x} p(x) \log q(x) \tag{1.9}$$

In this case as the target word is a one-hot vector, the distribution p in 1.9 is equal to zero except for the j-th component then

$$\begin{aligned} E(w_T, w_O) &= -\sum_{i=1}^{V} w_{T_i} \log w_{O_i} \\ &= -\log w_{O_j} \\ &= -\log p(w_j|w_{I_1}, ..., w_{I_C}) \end{aligned} \tag{1.10}$$

Finally using 1.7 and 1.8 one can define the loss function as

$$\begin{aligned} E(w_T, w_O) &= -\log p(w_j|w_{I_1}, ..., w_{I_C}) \\ &= -\log \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} \\ &= \log \sum_{j'=1}^{V} exp(u_{j'}) - u_j \end{aligned} \tag{1.11}$$

One way to minimize this loss function is to use the gradient descent technique or stochastic gradient descent (SGD) which allow to find a local minimum of a differentiable function or a sum of differentiable functions and then to reduce the error by updating the weight matrices accordingly.

Definition 5 (Gradient descent)**.** *It is a first-order iterative optimization algorithm for finding the minimum of a differentiable function. To find a local minimum of a function using gradient descent, the idea is to move step by step in the direction of the negative gradient (or of the approximate gradient) of the function at the current point. This technique works in spaces of any number of dimensions and even if it exists better alternatives this one stays computable for extremely large problems. Several extensions are also used such as the momentum method which reduces the risk of getting stuck in a local minimum.*

The Skip-gram model (Figure 1.4) is the opposite of CBOW but the equations are very similar. This model uses the same matrices W and W'. As there is only one input word, for the Skip-gram model the output of the hidden layer is simply a projection of the word w_I on a D-dimensional space using the weight matrix W and 1.7 becomes:

$$h = w_I \cdot W = W_{k,*} \tag{1.12}$$

The difference now with CBOW is that one need to compute C probability distributions and not only one. All of these distributions will be computed using the same weight matrix W' as following:

$$p(w_{O_j}|w_I) = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{1.13}$$

And then for the backpropagation, the loss function is changed to:

$$\begin{aligned} E &= -\log p(w_{O_1}, w_{O_2}, ..., w_{O_C}|w_I) \\ &= -\log \prod_{c=1}^{C} p(w_{O_c}|w_I) \end{aligned} \tag{1.14}$$

Or it exits $j_c \in [|1; V|]$ such as w_{O_c} is a one-hot vector with a 1 only in j_c-th position.

Hence:

$$\begin{aligned} E &= -\log \prod_{c=1}^{C} \frac{exp(u_{c,j_c})}{\sum_{j'=1}^{V} exp(u_{i,j'})} \\ &= C \cdot \log \sum_{j'=1}^{V} exp(u_{j'}) - \sum_{c=1}^{C} u_{j_c} \end{aligned} \tag{1.15}$$

This equation is very similar to 1.11, the computed probability distribution needs to match with the C context words that are known during the training. The same technique of gradient descent is also used for backpropagation in the Skip-gram model.

These two models don't seem to generate word vectors but here is the trick that Tomáš Mikolov, K. Chen, et al. 2013 used, after the training each line of the input matrix W will be in fact a very good representation of each words in a D-dimensional space. Indeed, if two different words have similar contexts, ie the same words appear around them, then the model needs to output very similar results. One way for the network to do this is to have similar word vectors for these two words. However as Goldberg and Levy 2014 argue, there is a strong hypothesis saying that words in similar contexts have similar meanings which should to be better defined even though as human we often use the context to determine the meaning of a new word.

Tomáš Mikolov, K. Chen, et al. 2013 compared these two models with the previous NNLMs (feedforward and recurrent) on semantic and syntactic questions. They concluded that, even with distributed frameworks to train the models, because of the much lower computational complexity it is possible to train the CBOW and Skip-gram models on several order of magnitude larger datasets than the other NNLMs. Thanks to this much better training CBOW and Skip-gram outperform more complex NNLMs on the tested NLP tasks with an advantage to the Skip-gram model which work well with "small" amount of training data and represent better rare words than the CBOW model which is several times faster to train and has a slightly better accuracy on frequent words.

Parametrization of the CBOW and Skip-gram models

With the objective to improve even more the Skip-gram model, Tomáš Mikolov, Sutskever, et al. 2013 proposed several useful additional techniques.

Hierarchical softmax As seen in Equation 1.13 of the Skip-gram model that uses the softmax function, to train the model we need to compute the gradient of the loss function which is proportional to the size of the vocabulary V as E is a V-dimensional function. The problem here is that V is often very large, between 10^5 and 10^7 terms so it becomes very time consuming. Hierarchical softmax is a computationally efficient approximation of the softmax and was introduced by Morin and Bengio 2005 and Mnih and Hinton 2009. The model uses a binary tree to represent all the words of the vocabulary at its leaves. Each inner node represents the relative probabilities of its child nodes. $p(w|w_I)$ is defined as follows:

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([|n(w,j+1) = ch(n(w,j))|] \cdot {v'_{n(w,j)}}^T \cdot v_{w_I}) \tag{1.16}$$

where $n(w,j)$ is the j-th node on the path from the root to the word w, $L(w)$ the length is this path, $ch(n)$ is an arbitrary child of the node n, $[|x|] = 1$ if x is true and -1 otherwise and $\sigma(x) = \frac{1}{1+\exp(-x)}$. Also we can notice that the hierarchical softmax need only one representation v_w of each word w (two representation per word for the regular softmax) but also a representation v'_n of every inner node of the binary tree. We can see on this formula that the cost of computing the gradient of $\log p(w|w_I)$ is proportional to $L(w)$ which is not greater than $\log W$ in average (by construction of the binary tree). Morin and Bengio 2005 showed that a significant speed-up of around 200 times faster can be achieved with only a little degradation of the performance of the model. Later Mnih and Hinton 2009 tried to construct this binary tree with different methods and showed the necessity to carefully construct the hierarchy over the words which impacts a lot the performance. Hence Tomáš Mikolov, Sutskever, et al. 2013 chose to use a Huffman tree which assigns short codes (and shorter paths in the tree) to the frequent words as it has been observed that grouping words together by their frequency is a very simple speed-up technique for NNLM (Tomáš Mikolov, Kombrink, et al. 2011, Tomáš Mikolov, K. Chen, et al. 2013).

Negative Sampling An alternative to the hierarchical softmax seen previously is the Noise Contrastive Estimation (NCE) introduced by Gutmann and Hyvärinen 2012 and applied to language modelling by Mnih and Teh 2012. NCE allows to approximately maximize the log probability of the softmax function. Tomáš Mikolov, Sutskever, et al. 2013 defined Negative Sampling (NEG) as a simplified version of the NCE, simplification possible due to the specific task of word embedding, and incorporated it to the objective

function in replacement of the hierarchical softmax.

According to the Tomáš Mikolov, Sutskever, et al. 2013, hierarchical softmax works better for infrequent words while negative sampling works better for frequent words and better with low dimensional vectors. As training epochs increase, hierarchical softmax stops being useful so the method of Skip-Gram with Negative Sampling (SGNS) appears to be a good choice of parametrization.

Subsampling frequent words The most frequent words like "in", "the" or "of" occur a lot of times and provide less information than the rare words. To avoid to loose time to train the vector representation of frequent words that will not change significantly after millions of training, Tomáš Mikolov, Sutskever, et al. 2013 used a simple subsampling approach to differentiate rare from frequent words. Each word w of the training set will be discarded with a probability equals to:

$$P(w) = 1 - \sqrt{\frac{t}{f(w)}} \tag{1.17}$$

where $f(w)$ is the frequency of w and t a chosen threshold usually around 10^{-5}. This subsampling technique accelerates the learning phase and even significantly improves the accuracy of the rare word vectors.

GloVe

Proposed by Pennington, Socher, and Christopher D Manning 2014, Glove, that stands for Global Vectors, is another log-bilinear model of word embedding that is trained on the global co-occurrence matrix using a weighted least square objective function. They derived this objective function from the ratio of co-occurrence probabilities that two terms have regarding a third pivot word. Indeed the raw probability P_{ik} that a term k co-occurs with a term i may not be enough to capture the relationship between the terms and a comparison point with another term j may give more information. It gives a basic model of the form:

$$F(w_i, w_j, w_k) = \frac{P_{ik}}{P_{jk}} \tag{1.18}$$

They also proved that the SGNS method has in fact an equivalent objective function than the one they use so the two methods have a similar approach but they concluded through extensive experiments that GloVe makes use of these statistics more efficiently than the

first word2vec methods. These methods evolved since then and it is not the case anymore.

fastText

No matter the word embedding technique seen so far, they represent each of the keywords as a distinct vector. However in morphologically rich languages, like French or Spanish where verbs have tens of inflected forms, they are many word forms that appear rarely in training corpus making them very difficult to represent as vector. These words are very often even not selected as keywords and we don't even try to learn their vector representation. With fastText, Bojanowski et al. 2017 proposed a way to improve the word representation of such rich languages by learning vectors for character n-grams. (Definition 2).

Unlike for the keyword extraction we consider here n-grams at the character level that is a word will be represented by the set of the possible sequences of n characters it contains. To highlight prefixes and suffixes the symbols $<$ and $>$ were added to the word before determining its character n-grams. As an example the word *where* which is represented by the sequence *<where>* and the character tri-grams *<wh*, *whe*, *her*, *ere* and *re>*.

As an extension of the Skip-gram model (presented in Section 1.1.3), fastText will learn representation of the character n-grams in a very similar way, trying to minimize the log-likelihood of sampled negative instances. As a word will be represented by the sum of the vector representations of its character n-grams, the scoring function 1.7 becomes

$$s(w_t, w_c) = \sum_{g \in \mathcal{G}_{w_t}} z_g^T v_{w_c} \tag{1.19}$$

where z_g is the vector representation of the character n-gram g of w_t.

One of the major features of this approach of word embedding is that it also allows to determine the representation of out-of-vocabulary words, i.e. that don't appear in the training corpus, by simply summing their character n-grams (if, of course, at least one of them appeared in the training data).

According to Bojanowski et al. 2017, fastText performs significantly better compared to CBOW when the size of the corpus is small but when the size of the corpus grows, as it is easier to find occurrence of rare words which will eventually lead to good representation using classical word embedding algorithms, the difference tends to be less important.

They also discuss the choice of character n-grams to compute. More sophisticated n-

grams could be considered instead of taking arbitrary sequences of n characters such as known suffixes, prefixes or stems. Indeed with a choice of values for n between 3 and 6, they showed that for inflected words in German, English and French, the most important n-grams used for a word representation are often valid morphemes and affixes.

1.2 Hierarchical clustering

As a reminder, epistemologists would like first to understand the existing structure between scientific domains before looking at their evolution. Then we present in this Section different possible clustering algorithms and especially hierarchical clustering algorithms that is logically the first step of our workflow.

Clustering is the task of grouping objects by sets such that objects that belong to the same group are more similar to each other than to objects of other groups. Such groups are called clusters.

A distinction can be made between hard and soft (or fuzzy) clustering. *Hard clustering* assigns each object to at most one cluster when *soft clustering* produces for each object a membership probability distribution over all the clusters.

Flat clustering determines a set of clusters without any explicit structure or relation between them whereas *hierarchical clustering* produces an informative hierarchy of the clusters. Hierarchical clustering does not require to specify the number of clusters but it comes with a greater complexity (at least quadratic) compared to flat clustering algorithms having a linear complexity such as *k-means* and *expectation-maximisation algorithms*.

Hierarchical clustering algorithms can be either *top-down* or *bottom-up*. The *bottom-up* type is agglomerative, it considers each object as a singleton cluster and iteratively merge clusters until all of them are reunited into a single cluster containing all the objects. *Top-down* algorithms are divisive, using a cluster splitting method they proceed in the opposite way than agglomerative clustering by splitting clusters until individual objects are reached.

Typical representation of hierarchical clustering is a *dendrogram* (Definition 6) as shown in Figure 1.7a but a *Venn diagram* shown in Figure 1.7b is another possibility.

Definition 6 (Dendrogram)**.** *From the Greek dendron that means tree and gramma that means draw, a dendrogram is the typical representation of the result of a hierarchical clustering algorithm. The bottom layer that contains the leaves represents the singleton*

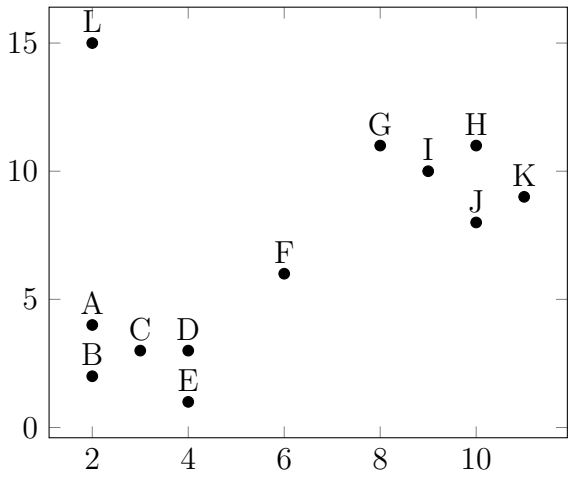


Figure 1.6: An example of 12 two-dimensional points to cluster.

clusters and the top layer contains one cluster of all the objects. Moving up or down in the dendrogram allows to reconstruct the history of the merges.

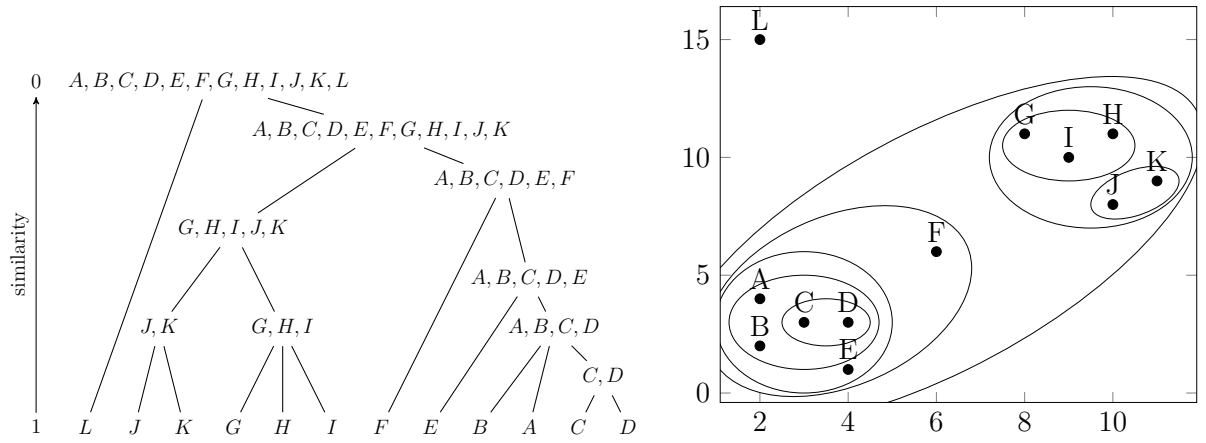
Formally, a dendrogram over a finite set E is a mapping function θ from $[0, 1]$ to the set of partitions of E verifying:

— $\theta(0)$ *is the set of singletons of* E

— $\exists t_0, \forall t > t_0, \theta(t) = \{E\}$

— $\forall t < t', \theta(t)$ *is a refinement of* $\theta(t')$*, i.e.* $\forall C \in \theta(t), \exists C' \in \theta(t')$ *such as* $C \subset C'$

For some tasks one can want to retrieve a set of clusters that cover the data (which is a partition if the clusters are disjoint) similarly to the result of a *flat clustering*. To this purpose it exists several methods to flatten a dendrogram. One can use an arbitrary criterion such as cutting the dendrogram at a given similarity value. For instance with a chosen similarity at 0.6, it will produce a set of clusters that have a pairwise similarity lower than 0.6. For a similarity of 0.4 it will produce less clusters than for a similarity of 0.6. Indeed we cut the dendrogram at a higher level as the similarity decreases from the leaves to the root (see Figure 1.7a). In the same way, one can cut the dendrogram for a chosen number of clusters independently to the value of similarity. Other methods consists in finding automatically the right cutting point based on a quality criterion. For instance cutting the dendrogram where the similarity gap between two levels is the largest seems a good possibility as the internal quality of the clusters will decrease the most when merging more clusters. Similar techniques exist for other types of clustering when it comes to guess the correct number of clusters.



(a) An example of dendrogram. (b) An example of Venn diagram.

Figure 1.7: Possible representations of hierarchical clustering on the example of Figure 1.6.

Hierarchical agglomerative clustering is the most frequent in the literature and in the following we will see different approaches and discuss their strengths and weaknesses.

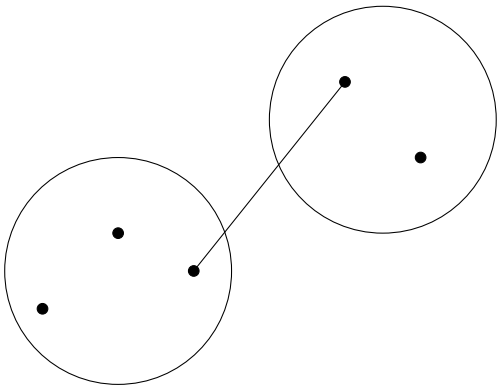
1.2.1 Hierarchical agglomerative clustering (HAC)

One of the simplest and naive method of HAC consists in computing the $N \times N$ similarity matrix S and then in $N - 1$ steps merging the current most similar clusters. At each step the rows and columns of the merged clusters are removed and new ones are added for the new cluster. These new row and column are filled with its similarity to the other remaining clusters. The method of computing the cluster similarity is at upmost importance as it is at the heart of this HAC algorithm.

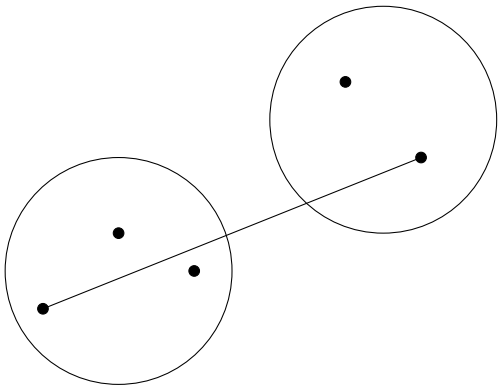
Similarities and distances are correlated. If they is no single definition of a similarity measure, usually such a measure is the inverse of a distance metric. The similarity measure takes large values for similar objects and reaches zero for very dissimilar objects hence objects having a small distance are similar whereas further objects are less similar. This distance metric is then a *dissimilarity measure*.

Linkage criteria

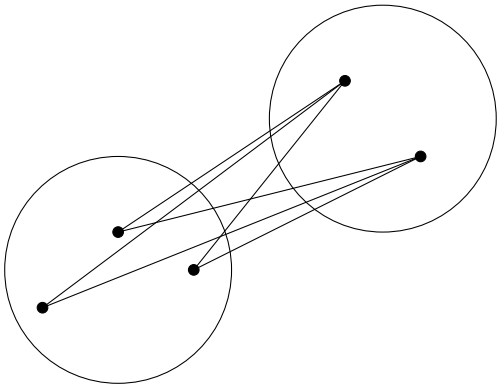
Assuming we have a metric d computing the elementary distance between any two objects of X, we can define a linkage criterion D that determines the distance between any subsets C_i and C_j of X in several ways.



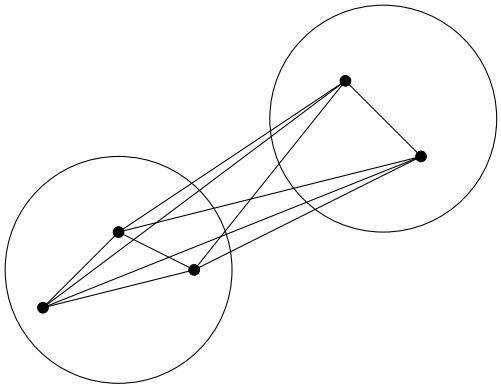
(a) Maximum similarity (single-linkage).



(b) Minimum similarity (complete-linkage).



(c) Average inter-similarities (centroid-linkage).



(d) Average of all similarities (group-average).

Figure 1.8: Visualization of different linkage functions for HAC algorithms (adapted from Christopher D. Manning, Raghavan, and Schütze 2009).

Single linkage In *single linkage clustering* the distance between two clusters is the distance of their closest elements (Figure 1.8a).

$$D(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} d(x_i, x_j) \tag{1.20}$$

The attention is given to only a local part of the clusters where they are the closest hence this method is also known as *nearest neighbour clustering*.

Complete linkage In *complete linkage clustering* the distance between two clusters is the distance of their farthest elements (Figure 1.8b).

$$D(C_i, C_j) = \max_{x_i \in C_i, x_j \in C_j} d(x_i, x_j) \tag{1.21}$$

If the attention is also given to a local part of the clusters, at the opposite of the *single linkage algorithm*, this method is also known as *farthest neighbour clustering*.

Chaining effect and outliers Based on only a pair of cluster elements, *single-linkage* and *complete linkage clusterings* cannot reflect a great diversity of clusters. *single-linkage clustering* tends to produce elongated clusters, a chain of points can be extended for long distance no matter the global shape of the cluster. This is called the *chaining effect* and is identified as one of the main drawback to this type of clustering. On Figure 1.7a the point F is responsible on a chaining effect between the clusters $\{A, B, C, D, E\}$ and $\{G, H, I, J, K\}$. On the opposite, *complete linkage clustering* pays too much attention to outliers, i.e. to isolated points that gravitate around a cluster compared to another close dense cluster. The points on Figure 1.6 will produce the clusters $\{A, B, C, D, E, F\}$, $\{G, H, I, J, K\}$ and $\{L\}$ at some point because $d(B, F) < d(F, H)$. Then $\{L\}$ will be merged with $\{G, H, I, J, K\}$ as $D(\{G, H, I, J, K\}, \{L\}) = d(L, K) < D(\{G, H, I, J, K\}, \{A, B, C, D, E, F\}) = d(B, H) < D(\{A, B, C, D, E, F\}, \{L\}) = d(E, L)$. More importance is given to the singleton cluster $\{L\}$ than for *single-linkage clustering*.

Centroid linkage In *centroid linkage clustering* the distance between two clusters is the distance of their centroids.

$$\begin{aligned} D(C_i, C_j) &= d(c_i, c_j) \text{ where } c_i \text{ (resp. } c_j) \text{ is the centroid of } C_i \text{ (resp. } C_j) \\ &= d\left(\frac{1}{|C_i|}\sum_{x_i \in C_i} x_i, \frac{1}{|C_j|}\sum_{x_j \in C_j} x_j\right) \\ &= \frac{1}{|C_i| \times |C_j|}\sum_{x_i \in C_i}\sum_{x_j \in C_j} d(x_i, x_j) \end{aligned} \tag{1.22}$$

As the distance of the sum of vectors is equal to the sum of their distances, Equation 1.22 shows that the centroid distance is equivalent to the average sum of distances between elements of C_i and C_j (Figure 1.8c). Unlike *single linkage clustering* and *complete linkage clustering* this variant considers equally each point of the clusters.

Group-average linkage In *group-average linkage clustering* the goal is to produce coherent clusters. The resulting cluster of a merge should have a high pairwise similarity of its elements (Figure 1.8d). There are $|E| \times (|E| - 1)$ possibilities to choose two elements in a set E ($|E|$ possibilities to choose the first element and $|E| - 1$ possibilities for the second one). Hence the distance used to merge clusters is as defined in Equation 1.23.

$$D(C_i, C_j) = \frac{1}{(|C_i| + |C_j|) \times (|C_i| + |C_j| - 1)}\sum_{x_k \in C_i \cup C_j}\sum_{x_l \in C_i \cup C_j, x_l \neq x_k} d(x_k, x_l) \tag{1.23}$$

Ward's method In fact as Ward Jr 1963 mentioned, any objective function chosen by the investigator could be used for the hierarchical agglomerative clustering. They proposed to use an objective function based on the *error sum of squares* also known as the *Ward's minimum variance method*. At each step the clusters that are merged are those that lead to a minimum increase in within-cluster variance. In the python package *scipy*[3] that implements this method, it means that the distance between a new formed cluster u that

3. `https://www.scipy.org/`

merges s and t and another cluster v is given by:

$$D(u,v) = \sqrt{\frac{|v|+|s|}{|v|+|s|+|t|}D(v,s)^2 + \frac{|v|+|t|}{|v|+|s|+|t|}D(v,t)^2 - \frac{|v|}{|v|+|s|+|t|}D(s,t)^2} \tag{1.24}$$

We need to note that some linkage methods are correctly defined only with specific metrics. That is the case for instance for the *centroid* and *Ward* methods that requires to use the Euclidean distance.

Complexity

Classical HAC algorithms have a time complexity of $\mathcal{O}(n^3)$ and a memory complexity of $\mathcal{O}(n^2)$. If this time complexity was reduced to $\mathcal{O}(n^2 \log n)$ using a special data structure (a heap), it came with an increased memory requirements. Also optimal efficient methods in terms of time complexity in $\mathcal{O}(n^2)$ have been found for *single linkage algorithm* with *SLINK* (Sibson 1973) and for *complete linkage algorithm* with *CLINK* (Defays 1977).

1.2.2 Density-based clustering

In density-based clustering clusters are defined as regions in the data that have a higher density. The elements in the sparse areas that are required to separate clusters are considered as noise or border points.

DBSCAN One of the most widely-used algorithms of this category is *DBSCAN* proposed by Ester et al. 1996. *DBSCAN* stands for *Density-Based Spatial Clustering of Applications with Noise*. This method connects data points that satisfy a specific density-based criterion: the minimum number of other data points within a given radius must be above a predefined threshold.

Formally, considering two parameters ϵ, the radius of the element neighborhoods, and $minPts$, a lower bound of the number of elements required to form a dense region, *DBSCAN* classifies elements as follows:

— If at least $minPts$ fall in the ϵ-neighborhood of an point p then p is considered as a *core point*.

— If a point q is within a distance ϵ of a core point p then q is *directly density-reachable* from p . If q is not itself a core point the opposite is not true and q is then considered as a *border point*.

— All points not directly density-reachable from at least one other point are considered as *noise points*.

Clusters are the defined by two properties:

— (Connectivity) All elements of a cluster are *density-reachable*. Two points p and q are density-reachable if it exists a sequence of points that are directly density-reachable between p and q.

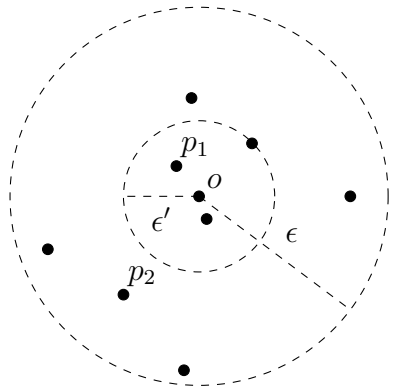
— (Maximality) Any point q directly density-reachable from a point p of a cluster C also belongs to this cluster.

The main advantage of this method is that it allows to detect clusters of arbitrary shapes which is not possible with technique such as *k-means* and that with a worse case complexity in $\mathcal{O}(n^2)$ which can be reduced to $\mathcal{O}(n \log n)$ using a tree-based spacial index. However it expects a clear density drop around the clusters which is not always the case with Gaussian distributions for instance. Also it cannot correctly detect clusters with large panel of densities due to the parameters $minPts$ and ϵ that specifically target clusters of a given density.

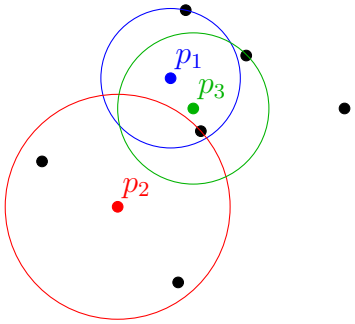
OPTICS Based on the same principle as *DBSCAN*, *OPTICS*, for *Ordering Points To Identify the Clustering Structure*, was proposed by Ankerst et al. 1999 and addresses the problem of detecting clusters with varying densities. It builds a spanning tree based on the *reachability distance*.

The reachability distance from a point p to a core point o is defined as the maximal value between the direct distance between p and o and the minimal radius ϵ' such as o still has $minPts$ points within this radius (and hence o is still considered as a core point). An illustration is given in Figure 1.9a.

Then by sorting the elements in a particular order, *OPTICS* builds a reachability bar plot, based on the reachability distance of the nearest neighbors, where clusters appear as valleys. This gives a hierarchical view of possible clusters according to the reachability distance on the y-axis.



(a) Reachability distances from o with $minPts = 4$:
- $d_{reach}(p_1, o) = core(o) = \epsilon'$,
- $d_{reach}(p_2, o) = d(p_2, o)$.



(b) Mutual reachability distances with $minPts = 4$:
- $d_{mreach}(p_1, p_3) = core(p_3)$ (as $core(p_1) < core(p_3)$),
- $d_{mreach}(p_2, p_3) = d(p_2, p_3)$.

Figure 1.9: Reachability distances used in *OPTICS* and *HDBSCAN**.

HDBSCAN* More recently improved versions of *DBSCAN* were proposed such as *HDBSCAN** (Campello et al. 2015). This new variant not only improved notions from *DBSCAN* and *OPTICS* but also proposed a procedure to extract a simplified cluster tree from the reachability relation which allows determining a hierarchy of the clusters.

Similarly to *OPTICS*, *HDBSCAN** uses a minimum spanning tree to build a hierarchy over the objects but using a different distance. While *OPTICS* in based on a reachability distance, *HDBSCAN** defined a slightly different *mutual reachability distance* as follows:

$$d_{mreach-minPts}(p, q) = max\{core_{minPts}(p), core_{minPts}(p), d(a, b)\} \tag{1.25}$$

It builds a tree structure similar to the one produced by a *robust single linkage clustering*. However they proposed a method to extract a flat clustering from this structure based on the notion of persistence and stability of a cluster. With $\lambda = \frac{1}{d}$, one can define λ_{birth} when a cluster is created from a split and λ_{death} when the cluster splits into smaller clusters. The persistence of a cluster is then given by $\lambda_{death} - \lambda_{birth}$, λ_{birth} appearing upper in the dendrogram than λ_{death}. The *stability* is computed from the sum of the persistence of each point composing the cluster as presented in Equation 1.26.

$$S(C) = \sum_{x \in C} (\lambda_x - \lambda_{birth}) \tag{1.26}$$

where $\lambda_x \in [\lambda_{birth}; \lambda_{death}]$ corresponds to the lambda value at which x falls out of the

cluster C that is λ_{birth} if x is still in C when it splits into smaller clusters or a greater value if x leaves the cluster earlier to be considered as a noise point. In order to extract a flat clustering from the tree structure, the idea is then to select the most stable cluster from each branch, i.e. with no overlap.

1.2.3 Graph-based clustering

A number of algorithmic methods have been proposed to identify communities in a graph (or network).

The first kind of methods produces a partition where a vertex can belong to one and only one community. Following the *modularity* function of Newman and Girvan 2004, numerous quality functions have been proposed to evaluate the goodness of a partition with a fundamental drawback, the now proved existence of a resolution limit which means that communities of different sizes inside a same network won't be correctly detected.

Definition 7 (Modularity)**.** *The modularity measures the quality of a partitioning of the nodes of a network of graphs. A good partitioning implies an important number of connections between nodes within a module but only few connections between nodes of different modules. Formally the modularity is defined as following*

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{1.27}$$

where:

- A_{ij} *is the value of the adjacency matrix between the vertices* i *and* j *(i.e. the weight of the edge between* i *and* j*),*
- k_i *(resp.* k_j*) is the sum of the weights of the edges adjacent to* i *(resp.* j*),*
- $2m$ *is the sum of the weights of all the edges of the network,*
- c_i *and* c_j *are the communities is witch the vertices* i *and* j *belong,*
- δ *is the Kronecker delta that is equal to* 1 *if* $c_i = c_j$ *and* 0 *otherwise.*

The modularity takes values between -1 *and* 1*. It is negative when the proportion of edges inter-communities is more important than intra-communities and conversely. It tends to* 0 *when the edges are uniformly distributed.*

Louvain method One of the leading methods based on this modularity is the *Louvain method for community detection* introduced by Blondel et al. 2008. It's an iterative

algorithm that tries to maximize the modularity in two-part steps.

First each node of the network is assigned to its own community (similarly to HAC algorithms). Then the first phase consists in computing the modularity changes if we remove a node i from its community and placing it in the community of each of its neighbors j. Then i is moved to the community of j that increases the most the modularity. If for all of its neighbors the modularity only decreases, i stays in its original community. This is repeated for each nodes until no improvement can be made (a node can be considered and moved several times) and the first phase ends. The order of the nodes processed has an importance on the new detected communities and at the end of this phase a local maximum of modularity is reached.

The second phase consists in building a new network by considering all detected communities at the first phase as a single node. The weight of the new edges between these communities is the sum of the weights of the edges between nodes of two communities. A self loop is also added to each community, weighted by the sum of the edges between nodes of this community.

The first phase can be reapplied until the set of communities does not change anymore which leads to the construction of a hierarchical structure inside each final community.

The second kind of methods, such as *CLIQUE* (Agrawal et al. 2005), *k-clique* (Palla et al. 2005), *DBLC* (Zhou et al. 2017) or *NMF* (Yang and Leskovec 2013), aims at finding sets of vertices that respect an edge density criterion which allows overlaps but can lead to incomplete cover of the network. Similarly to *HCOSM* (Qu et al. 2007), the method *EAGLE* (Shen et al. 2009) builds a dendrogram over the set of predetermined clusters, here the maximal cliques of the network so overlaps can appear only at the leaf level. Coscia et al. 2014 have proposed an algorithm to reconstruct a hierarchical and overlapping community structure of a network, by hierarchically merging local ego neighbourhoods.

1.2.4 Overlapping clustering

Fuzzy clustering methods (Bezdek 2013) allow that certain data points belong to multiple clusters with a different level of confidence. This way, the boundary of clusters is fuzzy and we can talk about overlaps of these clusters. In our definition it is a different notion, a data point either does or does not belong to a specific cluster and might also belong to multiple clusters. While *HDBSCAN* is closely related to connected components of certain level sets, the clusters do not overlap (since overlap would imply the connectivity).

Same for the *Louvain method* that determines a partition of the community network.

Only few overlapping hierarchies have been proposed.

Pyramid A *Pyramid* (Diday 1984) is a hierarchical structure that aims to allow a possible overlap of the clusters according to a specific criterion. Formally, let be E a set of elements and and C a set f non-empty parts of E, C is a *Pyramid* if:

— $E \in C$,
— $\forall x \in E$, $\{x\} \in C$ (all singletons are in C),
— $\forall C_i, C_j \in C$, $C_i \cap C_j \in C$ or $C_i \cap C_j = \emptyset$,
— It exists an total order θ on E that is compatible with C.

A dissimilarity d and an order θ are compatible if and only if for all ordered triplets $x_i \theta x_j \theta x_k$ of E we have $d(x_i, x_k) \geq max\{d(x_i, x_j), d(x_j, x_k)\}$. θ and C are compatible if the clusters of C are intervals of θ. These clusters can have common elements.

In a *Pyramid*, the clusters are intervals. It allows overlap by definition but the order apply a strong constraint on the clusters.

Weak and k-weak hierarchies The *weak hierarchy* (Bandelt and Dress 1989) and *k-weak hierarchy* (Diatta 1997) extend the notion of *Pyramid*.

Let be E a set of elements and and C a set f non-empty parts of E, C is a *weak hierarchy* if:

— $E \in C$,
— $\forall x \in E$, $\{x\} \in C$ (all singletons are in C),
— $\forall C_i, C_j, C_k \in C$, $C_i \cap C_j \cap C_k \in \{C_i \cap C_j, C_i \cap C_k, C_j \cap C_k\}$.

Let be E a set of elements and and C a set f non-empty parts of E, C is a *k-weak hierarchy* if:

— $E \in C$,
— $\forall x \in E$, $\{x\} \in C$ (all singletons are in C),
— $\forall C_1, C_2, \cdots, C_{k+1} \in C$, $\cap_{i \in \{1,\cdots,k+1\}} C_i \in \{\cap_{i \in \{1,\cdots,k+1\}-j} C_i | j \in \{1, \cdots, k+1\}\}$.

They are defined on the same principle, in *weak hierarchy* the intersection of any three clusters is the intersection of two of them and in *k-weak hierarchy* the intersection of k clusters is the intersection of $k-1$ clusters.

1.2.5 Clustering evaluation

As our workflow requires to perform clustering, it implies that we also need to evaluate the result of this clustering. Evaluation of clustering is as difficult as the clustering itself. It exists many methods that can be categorized in two main categories, *internal evaluation* where the evaluation is based on the clustered data themselves (coherence between cluster elements and/or how well the clusters are separated) and *external evaluation* where the evaluation is based on a given ground truth or gold standard. Other methods imply *manual evaluation* where clusters are verified by a human expert or *indirect evaluation* where the importance is given to the usefulness of the clusters for a given task.

Internal evaluation

Internal evaluation usually assigns the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity between clusters. One of the main drawbacks of this evaluation is that this evaluation is biased towards algorithms that will be based on the same cluster model. More, it is based on the assumption that a given cluster structure exists in the data and a clustering algorithm that performs well according to an evaluation method does not imply that this algorithm produces more valid results than another. An algorithm designed for some kind of models has no chance if the dataset contains a radically different set of models, or if the evaluation measures a radically different criterion. For instance, on a dataset with non-convex clusters neither the use of *k-means* that produces convex clusters, nor the use of an evaluation criterion that assumes convexity, will be able to properly detect the real structure of the data.

Among all the possible methods used for *internal evaluation* we will present some of the most used.

Davies–Bouldin index For n clusters and with a average distance to their centroid c noted σ the Davies–Bouldin index (Davies and Bouldin 1979) is defined as:

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \tag{1.28}$$

A set of clusters with a low Davies–Bouldin index has a high intra-cluster similarity and a low inter-cluster similarity and are hence considered as a good clustering according to this index.

Dunn index Like the Davies–Bouldin index, the goal of the Dunn index (Dunn 1974) is to identify dense and well separated clusters. On a set of n clusters it is defined as:

$$D = \frac{\min_{1 \leq i < j \leq n} d_{inter}(C_i, C_j)}{\max_{1 \leq k \leq n} d_{intra}(C_k)} \tag{1.29}$$

where the intra and inter-cluster distances are left to the user which left many possibilities. With this index, a set of clusters with a high value is more desirable.

Silhouette score The silhouette score (Rousseeuw 1987) measures how well an object fits to its own cluster compared to other clusters. For each object the silhouette score will take a value between -1 and 1, -1 indicating that the object should belong to another cluster and 1 that it really matches its cluster compared to the others. For an object $x_i \in C_i$ the average distance of x_i to the other objects of its cluster is given by:

$$a(x_i) = \frac{1}{|C_i| - 1} \sum_{x_j \in C_i, j \neq i} d(x_i, x_j) \tag{1.30}$$

We will compare this average distance to the minimum of average distances of x_i to elements of another cluster C_k. It can be computed as:

$$b(x_i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{x_j \in C_k} d(x_i, x_j) \tag{1.31}$$

The consideration is here given to the neighbour cluster of x_i where it would have more chance to fit if x_i wasn't an element of C_i. Finally the silhouette score is defined as follow:

$$s(x_i) = \begin{cases} \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}} & \text{if } |C_i| > 1 \\ 0 & \text{if } |C_i| = 1 \end{cases} \tag{1.32}$$

External evaluation

In *external evaluation*, clustering results are evaluated based on data that was not used for the clustering such as known class labels and external benchmarks. Having a ground truth on some data allow to count the number of time a pair of similar objects is assigned to the same cluster, the true positives (TP), and the number of time a pair of dissimilar objects is assign to different clusters, the true negatives (TN). An error is made when similar objects fall in different clusters, the false negatives (FN), or when

dissimilar objects are assigned to the same cluster, the false positives (FP). Like for *internal evaluation* it exists many variants and we will present some of them.

Rand index The *Rand index* (Rand 1971) simply corresponds to the proportion of correct decisions made by the algorithm. It is expressed as:

$$RI = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.33}$$

F-measure The *F-measure* (Chinchor 1992) uses a parameter $\beta \geq 0$ to balance the importance of false negatives. Considering the *precision* $P = \frac{TP}{TP+FP}$ of a clustering algorithm that corresponds to the proportion of similar elements in each cluster and the *recall* $R = \frac{TP}{TP+FN}$ that is the proportion of similar elements that correctly fall in the same cluster, the *F-measure* is calculated as:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{1.34}$$

Jaccard index The Jaccard similarity or Jaccard index (Jaccard 1901) measures the similarity between finite sets and is defined as the size of the intersection divided by the size of the union of the sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{TP}{TP + FP + FN} \tag{1.35}$$

If A and B are both empty, $J(A, B) = 1$ by definition.

Hierarchical clustering evaluation

All *internal* and *external evaluations* detailed above are based on a flat clustering. If it can be applied to hard clustering and also to overlapping clustering, these evaluations are not meant for more complex structures such as dendrograms and in most tasks the hierarchical structure is flattened which avoids the problem which is not what we want.

In some cases a ground truth is available and an evaluation task consists in comparing the result of a hierarchical clustering algorithm to this baseline. For instance for their task of hierarchical clusterings of scientific domains and disciplines, Dias et al. 2018 could rely on a hierarchy produced by the experts. However if labelling some data for flat

clustering can be complex, determining a hierarchy over these data is even more complex and impossible to do in most of the cases.

Hierarchy comparison To our knowledge, there are only few works on direct hierarchy comparison. Fowlkes and Mallows 1983 proposed a method that consists in cutting the hierarchies based on n elements in $k = 2$ to $n - 1$ clusters that will be compared using a measure B_k based on a $k \times k$ matching matrix $M = (m_{ij})_{1 \leq i,j \leq k}$. This matrix is filled with the number of common elements between each pair of clusters from the set of k clusters extracted from both hierarchies to compare. B_k is then computed as follow:

$$B_k = \frac{T_k}{\sqrt{P_k Q_k}} \tag{1.36}$$

with:

$$T_k = \sum_{i=1}^{k} \sum_{j=1}^{k} m_{ij}^2 - n \tag{1.37}$$

$$P_k = \sum_{i=1}^{k} m_{i.}^2 - n \tag{1.38}$$

$$Q_k = \sum_{j=1}^{k} m_{.j}^2 - n \tag{1.39}$$

where $m_{i.}$ (resp. $m_{.j}$) corresponds to the sum of the i^{th} row (resp. j^{th} column) of M.

1.3 Evolutionary structures

Structuring information to study or highlight an evolutionary phenomenon is not always an easy task. Following the evolution of an information through time is similar to following the evolution of the species in biology but based on different criteria than the gene evolution and with a really shorter temporality. One can follow the evolution of an information over a couple of days, month or a decade to the maximum while the evolution of the species take millions of years. Nevertheless several types of evolutionary structures have been proposed to automatically follow the evolution of a topic or an idea.

1.3.1 Co-citation network

Some methods used a co-citation network of scientific publications to study the evolution of a knowledge domain (C. Chen 2004, Rosvall and Bergstrom 2010). Usually the dataset is divided in period of one year and a co-citation network is build for each period. Then C. Chen 2004 clean the co-citation networks using thresholds and pruning techniques before merging them into a complete co-citation network. They produce a map that allows to detect interesting structures such as hub of co-citations of pivot articles. Rosvall and Bergstrom 2010 applied a clustering algorithm on the co-citation network. Clusters are identified according to their significance that is the probability of a document to belong in a given cluster. Then to represent the changes over time they produce an alluvial diagram between two periods that connects clusters from each period with stream fields.

However studying only the co-citation network has some strong drawbacks. One of the most important is that scientific papers are not represented equally. Some articles have much more than their fair share of citations, some have less, and some have none at all. The number of citations depends not only on the quality of the paper but also on many underlying factors. For instance, success breeds success, a highly cited article is likely to receive more citations than a currently less frequently cited article. Other methods based on the content of the documents have been proposed to avoid this problem.

1.3.2 Language evolution

Dias et al. 2018 explored the relationship between scientific fields. Using the hand made categorisation made by experts in Web Of Science, they produced hierarchies of scientific domains based on two dissimilarity metrics. The first dissimilarity corresponds to a co-citation dissimilarity where scientific fields are similar if many papers from two different scientific fields co-cite each other. The second dissimilarity considers the language used in the two fields one want to compare. Based on a generalized Jensen-Shannon divergence of the word distribution from the compared scientific fields, it has robust statistical properties from information theory. The found that the language and citation relationships between disciplines are similar and substantially different from the expert classification. Then they studied the evolution the the language dissimilarity between scientific fields over the years. If it is not possible to detect merge and split of scientific fields as they are determined by the experts, it is possible to see if two fields are getting closer or not at the language

level. But their conclusion of their study showed that in the last 30 years the language of the scientific fields remains unchanged and stayed on average at the same distance from each other. They proposed to interpret this phenomenon by the specialisation of science.

1.3.3 TextFlow

Cui et al. 2011 created a tool called *TextFlow* that produces topic flows (similar to alluvial diagrams from Rosvall and Bergstrom 2010) where topics are set of documents extracted from each period using Hierarchical Dirichlet process. They added a layout on this graph to highlight some keyword co-occurrences at specific times. It only represent the relationship between few selected key words based their weight in a cluster of documents represented by a variant of the TF-IDF measure.

1.3.4 Metro map

Shahaf, Guestrin, and Horvitz 2012 proposed to generate *Metro maps* that are concise structured sets of documents, one per node, maximizing coverage of salient pieces of information. In addition, the maps make explicit the various ways each piece relates to the others.

Definition 8 (Metro Map, Shahaf, Guestrin, and Horvitz 2012)**.** *A metro map M is a pair (G, Π), where $G = (V, E)$ is a directed graph and Π is a set of paths in G. We refer to paths as metro lines. Each $e \in E$ must belong to at least one line.*

In practice, each vertex of the Metro map is a document while the edges are based on the notion of *coherence* in a chain of documents. The coherence is defined by the *importance* of the common words that are related to each pair of documents of the chain. *Importance* is based on the well known *TF-IDF* statistics (Section 1.1.2). A global *coherence* can be determine as the minimal *coherence* among all the lines Π of the map.

They measured the quality of a metro map with the notions of coverage, diversity and connectivity. Formally, given a set of documents D, their goal is to find a map $M = (G, \Pi)$ over D which maximizes the connectivity and such that the coverage and the coherence of M are greater than a given threshold.

Later Shahaf, Yang, et al. 2013 proposed an improved version of the metro map with slight different properties:

— Metro stops are now represented as set of words instead of a single document.

— Metro stops being sets of words it allows zooming on a given topic to have a finer or broader overview.
— The number and the length of the metro lines are now not bounded by an input parameter.
— This new approach allows a better scalability of the proposed solution.

The measure of quality is similar to the one used for the first metro map definition but they added the notion of cluster, i.e. set of words, quality.

1.3.5 Phylomemetic structure

Chavalarias and Cointet 2013 proposed to build phylomemetic structures by analogy with the phylogenetic trees in biology. They start from a co-occurrence graphs of key terms of documents from different periods of time on which they perform a clustering to determine interesting topics. Then the inter-temporal matching consists in finding the best corresponding set of clusters in the target graph for each cluster of the source graph. This can be formulated by the following equation:

$$\Phi(C_i^T) = \{C_j^{T'}\}_{j \in \kappa_i^T}, \kappa_i^T = argmin_{\kappa \subset K^{T'}} d(C_i^T, \cup_{k \in \kappa} C_k^{T'}) \tag{1.40}$$

A threshold can also be applied to the process by stopping a branch of evolution if the best union of clusters in the target is too different from the source cluster and then considered as unsatisfactory.

Chapter 2

Overlapping Hierarchical Clustering (OHC)

2.1 Introduction

We consider the real life scenario where philosopher of Science would like to understand the structure of specific scientific domains based on a large corpus of publications, such as dblp or Wiley.

First we need to define the notion of scientific domains. We assume that a scientific domain or topic can be defined by a set of related key terms. It raises several issues:

— How can we efficiently extract scientific key terms from scientific papers?
— How can we determine the membership of a key term to a scientific domain, i.e. the relationship between terms? What about multi-domain key terms?
— How can we build a hierarchy from sets of terms?

The contribution developed in this Chapter in an answer of the last of these tree questions but first we need to discuss the choices we made to answer the two first ones based on the state of the art on the subject seen in the previous Chapter. More precisely we will explain our choices to transform a text into embeddings in Section 2.2 and present important definitions in Section 2.3. Then we will present our new OHC algorithm in Section 2 and some properties in Section 2.5. We will finally propose an analysis of this algorithm on different aspects in Section 2.6.

2.2 From text to embeddings

2.2.1 Text formatting

Time period selection

The structure of scientific domains is closely related to the period we are looking at. Indeed the relations between the domains and the theories they contain can be completely different between two distant moments in time. Scientific breakthroughs not only changed the leading theories of a domain but also its relation with other domains. It is even more the case nowadays considering the increase of inter-disciplinary projects, if a domain evolves for some reason, it will impact not only scientists from the domain but also all of those who use techniques of this domain. For instance, if a team of chemists discover a new method to produce hormones, it will have an impact on ethologists that study animal behaviour even if they are not considered working in the same area.

Hence to build a hierarchical structure of scientific domains we need to select papers from a given period of time. It is reasonable to say that scientific papers published at a given period represent the state of the art of the domains they cover. It will contain improvements of existing theories or present new approaches challenging the theories in place. If there is no longer published paper on a given theory then we assume that this theory is dead, either having been proven wrong or replaced by a better one. However, the length of the period to consider is really significant. Science needs time and if we select a too short time window we won't be able to reconstruct a complete structure of the scientific domains as we will only detect the few domains that evolved in this short period. On the opposite if we select a too wide period we will miss all the evolutions that occur during this period. We need to have in mind that the final goal is to study the evolution of the scientific domains then we need to find a good balance between having the most complete map of science for a given period without occulting the evolution detection. It also depends on the level of details we would like, if we have enough data and we want to retrieve only the big revolutions that appeared in the history of science evolution, the passage from the classical mechanics to the general relativity for instance, we will of course select wider periods of time. If we want to have a closer look to a specific domain, selecting the proper time window will also depend on this domain, some evolves quickly and will require a small window when others need more time and will require a larger window.

As we will use publicly available corpus of scientific paper such as Wiley or dblp (Section 2.6.1) that cover scientific papers from the last 20 to 40 years we will set the time window to 3 years. Also to avoid arbitrary cuts in the data that could appear in the middle of a particular event and make it undetected when we will be interested of the evolution of the domains, we won"t take distinct periods of time, i.e the 3 first years and the 3 next and so on and so forth. But we will rather slide the time window over the data with a given step, one year in our case, i.e. selecting the years 1 to 3 and then 2 to 4 and so on and so forth. The overlap on the periods will allow to detect interesting events as far as possible.

Data cleaning

Inspired by Dias et al. 2018 and detailed in Section 1.1.1 we will recap here the methods used to clean and format the data. We use the same techniques for the different datasets to be able to compare the results.

The title and the abstract when available were concatenated for each article to form a unique entry. Then we cleaned the text by converting it to lower case and replacing contractions by their non-contracted form.

At first we wanted to keep all inflected forms to avoid bias introduced by too much preprocessing and to have the best possible cover, but it lead to unnecessary computation at the bottom part of the hierarchy construction so we decided to perform a lemmatization algorithm. This will also allow to have a better diversity if we set the number of keywords to a given value, indeed having inflected forms will prevent us to detect as much canonical families as possible. For this reason we decided to use the English lexical database WordNet (Fellbaum 2017). The WordNet database contains more than 150 000 English words divided in four lexical categories, nouns, verbs, adjectives and adverbs. It ignores prepositions, determiners and other function words. WordNet also contains many types of relation between words including synonyms, hyponyms, and meronyms. It is a reference for word sense disambiguation (WSD), a task that aims to assign the context-appropriate meaning to words in a text. In general, knowledge-based approaches seem to be the most promising compared to learning approaches, supervised and unsupervised, due to their continuously enriched resources in front of the not available training corpora required for learning approaches (Navigli 2009). WSD is closely related to the lemmatizing task as a good lemmatizer needs to properly determine the meaning a word to determine its lemma. Thus WordNet is provided with, among many others, morphological functions that aim

to reduce a word to its lemma. Due to its rich structure, irregular forms such as *ate* are associated to their correct lemma, *eat* for our example. Of course WordNet is not perfect and has its limitations, in particular is has more difficulties to detect semantic relations between abstract concepts like emotions than concrete concepts. WordNet is also less efficient on domain-specific terminology.

However as explained in Section 1.1.1 the part of speech (POS) associated to a word in a sentence is at upmost importance to determine the correct canonical form of the word. Hence to help the WordNet lemmatizer we use the Penn Treebank, the most popular "tag set" for part of speech tagging for English (Marcus, Santorini, and Marcinkiewicz 1993). It is able to tag words in more than 36 categories and recognizes 12 different punctuation categories. To properly tag a word, it uses a stochastic algorithm based on the word position in the sentence. As the WordNet lemmatizer uses only four different part of speech categories, we convert specific Penn Treebank POS tag like "VVN" for a verb in the past participle to the general category "v" for verb understood by the lemmatizer. If a POS tag as no counterpart in the WordNet scheme we set it to "n" for noun which is the default category.

To sum up, we split each document in sentences, pass it to the Penn Treebank tagger and recover the POS tag of each word. Then we use the WordNet lemmatizer to determine the canonical form of the words based on their POS tag.

2.2.2 Word embedding

The first step of the workflow is complete and the text data are formatted in a proper way. For the second step we need to determine a way to select the correct keywords and keyterms in the corpus and answer the first of our questions. Now we see a corpus of documents as a subset of the entire corpus, subset containing only documents from a given period of time.

Keyterm extraction We saw many possibilities to extract keyterms a corpus of documents in Section 1.1.2. We concluded that interesting keyterms are not only words but also groups of words called *n-grams*. Hence from the lemmatized documents we transformed words to 3-grams when they appeared frequently enough together. We chose not to detect n-grams longer than three words. Indeed if 2-grams and 3-grams are quite frequent in English, it is less the case for longer sequences and it will be a waste of computational resources. Also we used a list of stop words (extremely common words like *of* or *the*)

to ignore when building the n-grams. This way sequences like "bank of america" will be directly detected as 2-gram. This list is a concatenation of several lists of English stop words from different sources, *gensim*[1], *nltk*[2] and *scikit-learn*[3] that are data mining python packages.

To determine the proper extraction method to select these n-grams as keyterms we need to have the finality in mind. Our task is not to classify the documents or to find the topics they are talking about. Hence the widely used TF-IDF method does not seem the most appropriate as it tends to select terms specific to only few documents. If a specific term appears in all the documents it is still interesting for us to have it as a keyterm as we want a good representation of the domains and theories that are present in the corpus to be able to build a good map of science. Also we will embed the keyterms into a vector space to connect them together, having similar representations for similar terms. In the literature of state of the art word embedding techniques (Tomáš Mikolov, K. Chen, et al. 2013, Bojanowski et al. 2017) they only used a minimum threshold of occurrences to select a term and for comparison purpose they also set a maximal size of the vocabulary. Our keyterm selection will be based on the same principle with different maximal sizes of the vocabulary depending on studied corpus. Stop words will also be excluded of the vocabulary. It slightly differs from the method used by Chavalarias and Cointet 2013 where they removed terms evenly distributed in the documents and where they also asked experts to clean the vocabulary from its non specific terms which is not applicable for large corpus related to many scientific domains.

Embedding generation In introduction of this chapter we raised the question of determining the relationship between terms and words from text documents. To access the relationship between terms we first need to determine their meaning. A naive and manual approach consists simply in looking for this term in a dictionary but we can notice that in a dictionary for every terms there are several definitions depending on the sense given to the term. It's on us to determine the correct sense of the term to access the correct definition. However for words we don't know it's impossible to do and naturally we try to guest the sense and often also the meaning at the same time by looking at the context in which the term is used. This process happens during our whole life because this is how we learn a language, mother tongue or extra language. Through our listening and reading,

1. `https://radimrehurek.com/gensim/index.html`
2. `https://www.nltk.org/`
3. `https://scikit-learn.org/stable/index.html`

when a term we don't know occurs, we guest its meaning by the context. It can be for more common words during our youth but also latter for technical terms and even for new slang terms. We properly learn the meaning(s) of the term only after we were confronted to it an enough number of time to cross the contexts and have a fine definition. Someone can easily misuse a word if he caught its meaning only from few examples.

We already evoked word sense disambiguation (WSD), the automatic task of assigning the context-appropriate meaning to a word in the previous section when we explained the role of lemmatization. WSD is at the center of the natural language processing challenges and already Weaver 1955 argued that WSD for machine translation should be based on the co-occurrence frequency of the context words near a given target word. It was soon followed by the famous quote from the linguist John Rupert Firth:

« *You shall know a word by the company it keeps.* »

— John Rupert Firth

This idea that *linguistic items with similar distributions have similar meanings* is known in linguistics as the distributional hypothesis. This hypothesis paved the way to the field of statistical semantics that favored the use of linear algebra as computational tool and representational framework, i.e. the use of high-dimensional vectors (embeddings) to represent words and terms.

Several word embedding techniques are presented in Section 1.1.3. They can be categorized as probabilistic models (Globerson et al. 2007), neural networks (Tomáš Mikolov, K. Chen, et al. 2013, Pennington, Socher, and Christopher D Manning 2014, Bojanowski et al. 2017), dimensionality reduction on the word co-occurrence matrix (Lebret and Collobert 2013, Goldberg and Levy 2014, Li et al. 2015) and many variants.

According to Goldberg and Levy 2014, neural network based word embedding techniques are in fact a efficient way to factorize probabilistic approaches and according to Levy, Goldberg, and Dagan 2015 the Skip-Gram with Negative Sampling (SGNS), one of the word2vec methods from Tomáš Mikolov, K. Chen, et al. 2013 is a strong baseline. Even if SGNS is not always the best technique, it never really underperforms compared to other neural network approaches. As we don't focus our work on the improvement of word embedding generation we will use this technique to map our vocabulary in a high-dimensional vector space. We chose standard tuning detailed by Tomáš Mikolov, Sutskever, et al. 2013 and explained in Section 1.1.3 with in particular a fixed number of dimensions to 300 and a subsampling of frequent words.

Dataset \ Vocabulary size	100	200	500	700	1000	2000
dblp (4M entries/34.4M words)	9	12	18	30	33	43
wiley (1M entries/174.4M words)	112	168	199	220	244	276

Table 2.1: Embedding computational time in seconds depending on the dataset and requested size of vocabulary.

As expected, the embedding computational time presented in Table 2.1 increases when the the number of terms to embed increases. We can notice that the embedding time is function on the number of words in the dataset and not function on the number of entries in this dataset. The Wiley dataset contains less scientific papers than the dblp dataset but more terms in total so it takes mote time to compute embeddings on the Wiley dataset than on the dblp dataset.

2.3 Intuition and basic definitions

The vectors are a representation of the context of the terms of the vocabulary and, as said previously, like our understanding of a word, they become more precise when they appear more in the documents and present a more diverse context to the learning algorithm. Terms that appear in similar context will have similar vector representations. It is reasonable to say that it is the case for terms for a given domain of science. Hence detecting dense regions of the vector space will allow to detect domains of science. To build a hierarchy of the domains, the natural way consists in regrouping close dense areas of the vector space together, i.e. regrouping close domains in a bigger one. Same as for the terms, domains identified in the same region will be more similar than domains from opposite side of the vector space. It answers partially the last of our questions on how we can build a hierarchy from a set of terms.

However how do we determine the dense areas of the vector space? One can set an arbitrary density threshold but it has no practical meaning and we have no idea on the number of sets of terms it will produce. Similarly in general we have no idea on the number of domains that should be detected so we can't adapt a hypothetical density threshold to produce a predefined number of domains. The idea is then to build directly the hierarchy over the term vectors like a classic agglomerative clustering algorithm. Several methods are presented in Section 1.2.

The main constraint that lead to the necessity of this new approach is related to the problem of multi-domain terms. Indeed there is no reason that a term belongs to only one scientific domain because, as we said, many terms have several meanings and sometimes there is no dominating one that could clearly put a term in a specific domain. If the word *space* is used in *musicology* as a position on the stave bounded by lines, it is used in great majority for its sense in *physics* and *maths* but on the contrary, the word *cell* belongs quasi-equally with different meanings to the field of *biology* and to the field of *telecommunication*. It was not always the case and it strongly depends on the considered period of time. Language change is a large field of linguistics by itself and is not the subject of this thesis, at least not of this Chapter where we consider separately periods of several years which are too short to be subject to language drifts.

Let's consider two regions of the vector space related to two distinct domains and a term that is related to these two domains. As a unique vector is associated to each term we can see this vector "attracted" by the two regions. Each time the term appears in the data, the learning algorithm will "correct" its position in the vector space by moving it closer to the other embedded terms of the domain that appeared in the same contexts. Hence our term vector will come and go around a position of balance located between the two regions. The position will depend on the proportion of use of the term in the two domains.

If the term vector is almost equidistant to the two domains, it will be tricky for hard clustering algorithms. With a density-based algorithm chances are that it will be considered as noise as the inter-domain space is less dense. Even for hierarchical clustering it is an issue as it causes the well known *chaining effect* (Section 1.2.1), putting it arbitrary in one of the two domains before forcing them to merge. Sometimes the hierarchies are not built from scratch but over a set of predetermine clusters that could have overlaps but it is not an intrinsic property of the hierarchical algorithm which will still cause the chaining effect. This is the case for instance for *HCOSM* (Qu et al. 2007) or *EAGLE* (Shen et al. 2009). Fuzzy clustering can easily deal with the multi-domain terms by partially assigning them to each domain with a probability of membership. However in our case there is not reason to consider *cell* belonging more to *biology* than to *telecommunication*. It should belong to both.

We proposed (Jeantet, Miklós, and Gross-Amblard 2020) to solve these issues and enable the term to belong to the two domains by allowing an overlap between them in a context of hard clustering.

To summarize we would like a method:
— that builds a **hierarchy** over a set of term vectors,
— that is based on the **density** of the vector space,
— that allows **overlaps** between identified clusters.

In a nutshell, our method obtains clusters in a gradual agglomerative fashion and in a precise way. At each step, when we increase the neighbourhood of the clusters by including more interconnections, we consider the points that fall in this connected neighbourhood and we take the decision to merge some of them whenever they are connected enough to a cluster using a *density criterion* λ. Taking interconnections into account may lead to overlapping clusters.

2.3.1 δ-neighbourhood

More precisely, we consider a set $V = \{X_1, \ldots, X_N\}$ of N points in a n-dimensional space, i.e. for $X_i \in V \subset \mathbb{R}^n$ where $n \geq 1$ and $|V| = N$. In order to explore this space in an iterative way, we consider points that are close up to a limit distance $\delta \geq 0$. We define the δ-neighbourhood graph of V as follows:

Definition 9 (δ-neighbourhood graph)**.** *Let $V \subset \mathbb{R}^n$ be a finite set of data points and $E \subset V^2$ a set of pair of elements of V, let d be a metric on $\mathbb{R}^n$ and let $\delta \geq 0$ be a positive number. The δ-neighbourhood graph $G_\delta(V, E)$ is an undirected graph with vertices labelled with the data points in V, and where there is an edge $(X, Y) \in E$ between $X \in V$ and $Y \in V$ if and only if $d(X, Y) \leq \delta$.*

Property 1. *If $\delta = 0$ then the δ-neighbourhood graph consists of isolated points while if $\delta = \delta_{max}$, where δ_{max} is the maximum distance between any two nodes in V then $G_\delta(V, E)$ is the complete graph on V.*

On such graphs it exists interesting internal structures like connected subgraphs and cliques.

Definition 10 (Connected subgraph)**.** *In graph theory, a **connected subgraph** of an undirected graph is a subgraph in which any two vertices are connected to each other by paths. A **connected component** is a connected subgraph that is connected to no additional vertices in the supergraph, i.e. a maximal connected subgraph.* [4]

4. `https://en.wikipedia.org/wiki/Component_(graph_theory)`

Definition 11 (Clique)**.** *In graph theory, a* ***clique*** *is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete. A* ***maximal clique*** *is a clique that cannot be extended by including one more adjacent vertex.* [5]

2.3.2 Distance in high-dimensional space

We introduced a notion of distance to define the δ-neighbourhood graph and if in theory any distance function may be suitable, we need to discuss how such metric behaves in relatively high-dimensional space in particular in regards to the so called *curse of dimensionality*.

Bellman 1966 introduced the concept of *curse of dimensionality* that refers to various phenomena that occur when analyzing and organizing data in high-dimensional spaces but that not exist in low-dimensional spaces. With the rise of the machine learning era, the most common phenomenon is the fact that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. This implies that every distance function lose their usefulness which becomes problematic for any distance based task like classification, supervised learning, clustering and many others.

It has been extensively studied by Zimek, Schubert, and Kriegel 2012 that temperate the phenomenon and showed that only hold in the scenario where data are independent and identically distributed (i.i.d) on each dimension. It is not the case when attributes are correlated. In fact they highlighted more specific problems in high-dimensional data and some that directly concern our task are as follow:

— Concentration of Scores: the distances of i.i.d. distributed objects converge to a normal distribution with low variance (central limit theorem), leading to numerically similar distances.

— Noise attributes: in high dimensional data, a significant number of attributes may be irrelevant.

— Definition of Reference-Sets: vicious circle between appropriate subspace and neighbor determination.

— Bias of Scores: different subspaces produce incomparable scores.

— Interpretation and Contrast of Scores: the scores often no longer convey a semantic meaning.

5. `https://en.wikipedia.org/wiki/Clique_(graph_theory)`

— Exponential Search Space: the search space can no longer be systematically scanned.
— Data-Snooping Bias: given the large search space, a hypothesis can be found for every desired significance.
— Hubness: certain objects occur more frequently in neighbor lists than others.

Concretely for our embedding space, as the dimension does not have a semantic meaning, choosing a too high number of them with respect to the size of the vocabulary might lead to irrelevant vector attributes which will only add noise for our clustering task.

Also in regards to the way that the data and the distances tend to behave in high-dimensional spaces, i.e. data becoming sparse and distances becoming similar, we will stick with the settings leading to the good results in term of semantic tasks in the word embedding literature. In most of the literature (Tomáš Mikolov, Sutskever, et al. 2013, Bojanowski et al. 2017 for instance) the size of the embedding space is set to 300 dimensions and the metric mostly used is the cosine distance (Definition 12) over the classical Euclidean distance.

Definition 12 (Cosine distance)**.** *The cosine distance D_C comes from the cosine similarity S_C which is defined as the cosine of the angle between two non-zero vectors. The similarity can be derived from the Euclidean dot product as*

$$S_C(A, B) = cos(\theta_{A,B}) = \frac{A \cdot B}{||A|| \times ||B||}$$

And the cosine distance is often calculated from the cosine similarity as

$$D_C(A, B) = 1 - S_C(A, B)$$

but this is not a proper distance metric as it does not respect the triangle inequality property. To obtain a proper metric, one can use the angular distance AD_C instead.

$$AD_C(A, B) = \frac{cos^{-1}(S_C(A, B))}{\pi}$$

As the cosine function, and hence the cosine similarity, falls in $[-1; 1]$, the angular distance falls in $[0; 1]$.

This can be explained because when the number of dimensions increases, the noise increases as well due to the "noise attribute" problem. And the same perturbation applied on 4 dimensions as a greater impact on the Euclidean distance than when applied on only

2 dimensions which make this distance sensitive to noise in high-dimensional spaces.

More, for tasks such as document indexing or semantic similarity detection, the magnitude of the vectors depends on extraneous factors like word occurrences whereas the angle between vectors is more immune to it. It can be easily perceived when documents are represented as term occurrences. If a term appears more in a given document it doesn't necessary mean that it is more specific to this document, it can come from the fact that this document in much longer than others. It is less intuitive but it is the same for word embedding vectors, the magnitude of the vector is impacted by the frequency of appearance of the words in the documents.

Hence distance will refers to the cosine distance for the rest of the thesis unless explicitly stated otherwise. Also in my opinion, even if 300 dimensions seems already a very high-dimensional space compared to the 3-dimensional space that surrounds us and to which we are used, it is relatively low in data science and tasks like classification, machine learning and clustering. For instance, with a vocabulary of 1000 terms, a co-occurrence matrix already represents the terms with 1000-dimensional vectors. Also one can think of the billions of billions of parameters in complex neural networks which are of a completely different order of magnitude than our problem size.

2.3.3 Graph density

Let's focus on our clustering problem. Varying δ will allow to progressively extend the neighbourhood of the vectors to form bigger and bigger clusters. It will be like if we were zooming out the data to have a more and more general view. The clusters will be formed according to the density of a region of the graph.

Definition 13 (Density, Diestel 2005)**.** *The density $dens(G)$ of an undirected graph $G(V, E)$ is given by the ratio of the number of edges of G to the number of edges of G if it were a complete graph, that is, $dens(G) = \frac{2|E|}{|V|(|V|-1)}$. If $|V| = 1$, $dens(G) = 1$.*

A cluster is simply defined as a subset of the nodes of the graph and its density is defined as the density of the induced subgraph.

As illustrative examples, in Figure 2.1, $C1$ is a part of a connected component (Definition 10) of the graph and its density $dens(C_1) = \frac{2}{3}$ as one of the 3 possible links between its vertices is missing. C_2 is a clique (Definition 11) and even a maximal clique, hence its density $dens(C_2) = 1$.

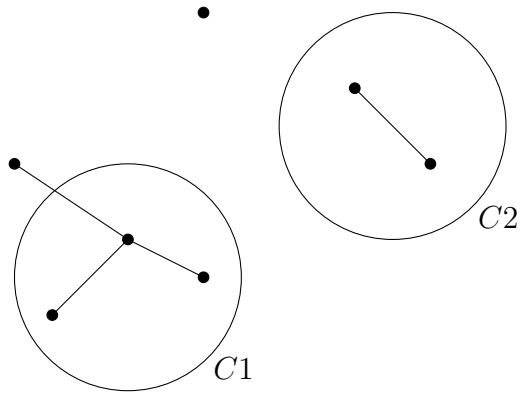


Figure 2.1: Example of clusters extracted from a δ-neighbourhood graph.

2.3.4 Quasi-dendrogram

Our algorithm, called OHC, computes a hierarchy of clusters that we can identify in the data. We call the generated structure a quasi-dendrogram and it is defined as follows.

Definition 14 (Quasi-dendrogram)**.** *A quasi-dendrogram is a hierarchical structure, represented as a directed acyclic graph, where the nodes are labelled with a set of data points, the clusters, such as:*

- *The leaves (i.e. the nodes with 0 in-degree) correspond to the singletons, i.e. contain a unique data point. The level of the leaf nodes is 0.*
- *There is only one root node (node with 0 out-degree) that corresponds to the set of all the data points.*
- *Each node (except the root node) has one or more parent nodes. The parent relationship corresponds to inclusion of the corresponding clusters.*
- *The nodes at a level δ represent a set of (potentially overlapping) clusters that is a cover of all the data points. Also, for each pair of points of a given cluster, it exists a path between points of this cluster that have a distance less than δ. In other terms, a node contains a part of a connected component of the δ-neighbourhood graph.*

An example of quasi-dendrogram can be found in Figure 2.2. For simplicity and comparison purpose, the distances should be normalized to be restricted to the interval $[0; 1]$. This is not mandatory and it depends on the distance used, for instance the cosine distance, by definition, already falls in $[0; 1]$. The main characteristic of the quasi-dendrogram and the property that makes it different from the classical dendrogram (Definition 6) is the *diamond shape* that can occur in the structure. It means that at some point a node N has at least two parent nodes P_1 and P_2 and that these parent nodes overlap at least

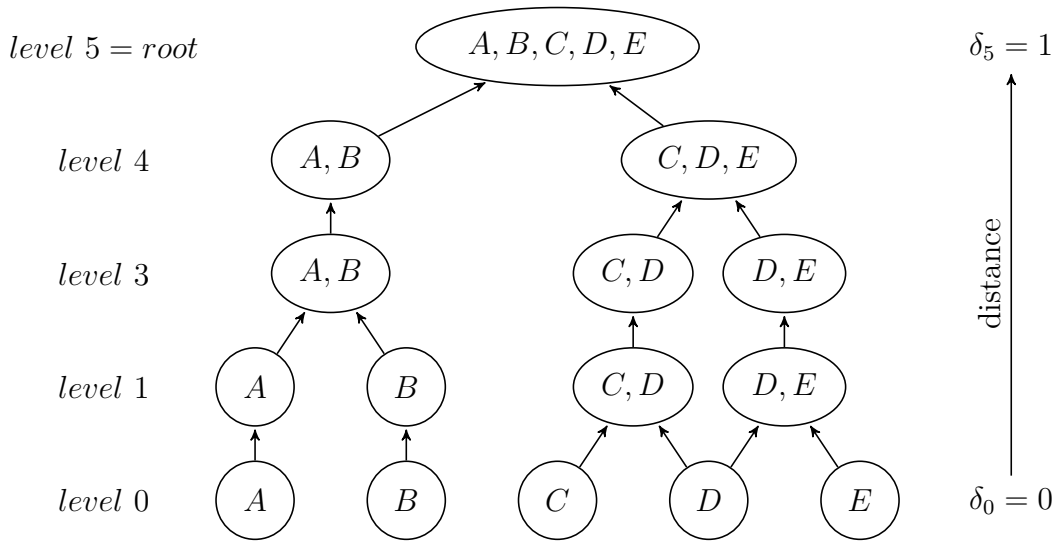


Figure 2.2: An example of **quasi-dendrogram** with its characteristic diamond shape.

on this child node. Formally we have $N \subseteq P_1 \cap P_2$. It appears on the Figure 2.2 in the right branch with the elements C, D, E. The singleton $\{E\}$ of the *level* 0 has two parents in the *level* 1, $\{C, D\}$ and $\{D, E\}$, and these 2 clusters will be merge in the *level* 4 to form the cluster $\{C, D, E\}$.

2.4 Overlapping Hierarchical Clustering (OHC) algorithm

The OHC method works as presented in Algorithm 1. We first compute the distance matrix of the data points (I3). We chose the cosine distance for reasons explained in Section 2.3.2. Then we construct and maintain the δ-neighbourhood graph $G_\delta(V, E)$, starting from $\delta = 0$ (I4).

We also initialize the set of clusters, i.e. the leaves of our quasi-dendrogram, with the individual data points (I4). At each iteration, we increase δ (I6) and consider the new added links to the graph (I8) and the impacted clusters (I9). We extend these clusters by integrating the most linked neighbour vertices if the density does not change more than a given threshold λ (I10-15). We remove all the clusters included in these extended clusters (I16) and add the new set of clusters to the hierarchy as a new level (I18). We stop when all the points are in the same cluster which means that we reached the root of the quasi-dendrogram.

Also to improve the efficiency of this algorithm we use dynamic programming to avoid to recompute information related to the clusters like their density and the list of their neighbour vertices. It lead to significant improvements in the execution time of the algorithm. We will discuss this further in the Section 2.6.

Algorithm 1 Overlapping Hierarchical Clustering (OHC)

Input:
— $V = \{x_1, \ldots, x_N\}$, N data points.
— $\lambda \geq 0$, a merging density threshold.
Output: quasi-dendrogram H.
Preprocessing: obtain $\Delta = (\delta_1, \ldots, \delta_m)$ the distances between data points in increasing order.
Initialization:
— Create the graph $G(V, E_0 = \emptyset)$.
— Set a list of clusters $C = [\{x_1\}, \ldots, \{x_N\}]$.
— Add the list of clusters to the level 0 of H.
$i = 1$.
while $\#C > 1$ and $i \leq m$ **do**
    for each pair $(u, v) \in V^2$ such as $d(u, v) = \delta_i$ **do**
        Add (u, v) to $E_{\delta_{i-1}}$.
        Determine the impacted clusters C_{imp} of C containing either u or v.
    for each impacted cluster $C_{imp_j} \in C_{imp}$ **do**
        Look for the points $\{p_1, \ldots, p_k\}$ that are the most linked to C_{imp_j} in G_{δ_i}.
        Compute the density $dens(S_j)$ of the subgraph $S_j = C_{imp_j} \cup \{p_1, \ldots, p_k\}$.
        if $S_j \neq C_{imp_j}$ and $|dens(S_j) - dens(C_{imp_j})| \leq \lambda$ **then**
            Continue to add the most linked neighbors to S_j the same way if possible.
            When S_j stops growing remove C_{imp_j} from the list of clusters C and add S_j to the list of new clusters C_{new}.
    Remove all cluster of C included in one of the clusters of C_{new}.
    Concatenate C_{new} to C.
    Add the list of clusters to the level δ_i of H.
    $i = i + 1$.
return H

2.5 Properties of the algorithm

Property 2 ($\lambda = 0$)**.** *When $\lambda = 0$, each level δ_i of a quasi-dendrogram contains exactly the maximal cliques (complete subgraphs) of the δ_i-neighbourhood graph G_{δ_i}.*

Property 3 ($\lambda = 1$)**.** *When $\lambda = 1$, each level δ_i of a quasi-dendrogram contains exactly the connected component of the δ_i-neighbourhood graph G_{δ_i}. In this case the OHC algorithm behave exactly like a single-linkage algorithm.*

Proof. To prove these properties, let us consider the comparison function $(C_1, C_2) \in C^2 \mapsto |dens(C_1) - dens(C_2)| \in [0, 1]$. It lands in $[0, 1]$ by definition of the density function $dens$ that also lands in $[0, 1]$. We prove our properties by induction on the level.

Initialization (level 0): by construction, the clusters of the level 0 of a quasi-dendrogram are the singletons and hence are the connected components (resp. cliques) of the 0-neighbourhood graph G_0.

General case: let us consider a level δ_i of a quasi-dendrogram where all the clusters are the connected components (resp. cliques) of the δ_i-neighbourhood graph G_{δ_i}. To determine the level δ_{i+1} we consider a new added link (u, v) such as $d(u, v) = \delta_{i+1}$.

For the $\lambda = 0$ case, as the current clusters are cliques, (u, v) cannot connects 2 vertices of the same cluster. So we suppose that (u, v) is a new link between the vertex v and the clique C_u, then v is added to C_u iff

$$\begin{aligned}
\text{(I13) is satisfied} &\iff |dens(C_u \cup \{v\}) - dens(C_u)| \leq \lambda = 0 \\
&\iff dens(C_u \cup \{v\}) - dens(C_u) = 0 \text{ by definition} \\
&\iff dens(C_u \cup \{v\}) = 1 \text{ as } C_u \text{ is a clique so } dens(C_u) = 1 \\
&\iff C_u \cup \{v\} \text{ is a clique.}
\end{aligned}$$

Hence, if v is added to C_u the clique property is preserved.

For the $\lambda = 1$ case, if (u, v) connects 2 vertices of the same cluster, then the connected component remains the same and the cluster is not changed. Otherwise, we suppose that (u, v) connects a new vertex v to the cluster C_u, that is to the connected component containing u which now contains a new vertex, v will always be added to C_u as $|dens(C_u \cup \{v\}) - dens(C_u)| \leq 1$ by definition so C_u becomes $C_u \cup \{v\}$ which is the new connected component. Hence v is added to C_u as soon as a link exists between them which preserves the connected component property. If the vertex v is also linked to other vertices, i.e. belong to the cluster C_v which is a connected component by recurrence hypothesis, the OHC algorithm will try to add the new adjacent vertices to $C_u \cup \{v\}$. These adjacent vertices are necessary those of the vertex v which are elements of C_v. For the same density reason v was added to C_u, any new adjacent vertex will be added to $C_u \cup \{v\}$. It will be repeated until no new adjacent vertex is found. It will be the case when the new formed cluster will be $C_u \cup C_v$. Indeed, as a connected component, it exist a path between any

two vertices of C_v and C_v is isolated from the rest of the graph. In fact with $\lambda = 1$ if two connected components are linked, they will be forced to merge. However the edges are added in increasing order to the graph so when new edge connects two clusters it will be an edge that connects the closest vertices belonging to different clusters of the graph. It corresponds to the smallest single-linkage distance (Section 1.2.1) which will make the OHC algorithm behave like a single-linkage algorithm.

Finally, the Properties 2 and 3 are proved by recurrence. □

2.6 Analysis

In this section we analyse the structure of quasi-dendrogram produced by the OHC algorithm. We will show different possible structures over a small example and on a real dataset. We will also discuss time complexity and few optimisation techniques.

2.6.1 Experimental setup

All our experiments are done on a Intel Xeon 5 Core 1.4GHz, running MacOS 10.2 on a SSD hard drive. Our code is developed with Python 3.5 and the visualization part was done on a Jupyter NoteBook. We used the *SLINK*, *average linkage* and *Ward* implementations from the scikit-learn python package and the *HDBSCAN** implementation of McInnes and Healy 2017.

Datasets

To partially see the scalability of our algorithm but also to avoid too long running times we had to limit the size of the datasets to few thousand vectors. To be able to compare the results, we run the tests on datasets of same sizes that we fixed to **100, 200, 500, 700 and 1000 vectors**.

— The first dataset is composed of **randomly** generated 2-dimensional points.

— To test the algorithm on real data and in our motivating scenario, the second dataset was created from the **Wiley** collection via their API[6]. We extracted the titles and abstracts of more than 1M scientific papers (174.4M words) and trained a word embedding model on the data of a given period of time by using the classical *SGNS* algorithm from Tomáš Mikolov, K. Chen, et al. 2013 following the

6. `https://onlinelibrary.wiley.com/library-info/resources/text-and-datamining`

recommendation of Levy, Goldberg, and Dagan 2015. We set the vocabulary size to a maximum of 1000 keyterms per period even though this dataset allows us to extract up to 50000 of them. This word embedding algorithm created 1000 300-dimensional vectors for each period of 3 years over the 20 years available with a sliding window of 1 year, creating 17 models in total.

— We built similar models using the historical data of the **dblp** collection (Hoffmann and Reitz 2018) that cover more than the 30 last years with 4M scientific papers (34.4M words). Despite the large amount of papers, the number of words is really low compared to the Wiley dataset. This is due to the lack of abstracts for most of the papers.

2.6.2 Expressiveness

With this small following example we would like to present the expressiveness of our algorithm compared to classical hierarchical clustering algorithms such as *SLINK* (Section 1.2.1). On the hand-built example shown in Figure 2.3a we can clearly distinguish two groups of points, $\{A, B, C, D, E\}$ and $\{G, H, I, J, K\}$ and two points that we can consider as noise, F and L. Due to the chaining effect we expect that a single-linkage algorithm will regroup the 2 sets of points early in the hierarchy while we would like to prevent it by allowing some cluster overlaps.

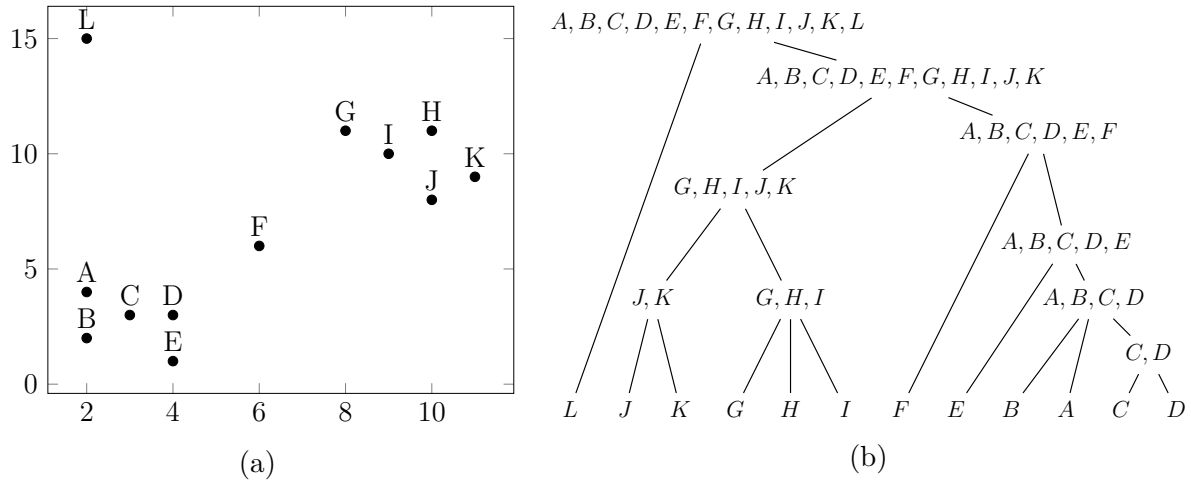


Figure 2.3: A hand-built example (a) and its single-linkage dendrogram (b).

Figure 2.3b shows the dendrogram computed by a single-linkage algorithm and we can see as expected that when F merges with the cluster formed by $\{A, B, C, D, E\}$ the next

step is to merge this new cluster with $\{G, H, I, J, K\}$.

On the contrary in Figure 2.4 that presents the hierarchy built with our method for a specific merging criterion, we can find the diamond shape that is specific to our quasi-dendrogram. For simplicity the view here slightly differs from the quasi-dendrogram definition as we used dashed arrows to represent the provenance of some elements of a cluster instead of going further down in hierarchy to have a perfect inclusion and respect the lattice-like structure. The merge between the clusters $\{A, B, C, D, E\}$ and $\{G, H, I, J, K\}$ is delayed to the very last moment and the point F will belong to these 2 clusters instead of forcing them to merge. Also depending on the merging criterion we obtain different hierarchical structures by merging earlier of later some clusters (see Figures in Appendix 2.8).

2.6.3 Flat cluster extraction

As mentioned in Section 1.2, certain tasks require a cover of the data formed by a set of clusters such as a flat clustering could produce. We saw that it is easy to extract a cover from the classical dendrogram using various methods. In this case we recall that the cover will be a partition of the data as they is no overlap between coexisting clusters.

If it is interesting to have a complete view of the hierarchy, it becomes impossible when the size of the data increases as the hierarchy will be too wide and too deep. By construction, quasi-dendrograms are even deeper than classical dendrograms due to the possible cluster overlaps that not necessary reduce the number of clusters at each level. The dendrogram depth is bounded by the number n of elements to cluster while the quasi-dendrogram depth is bounded by the number of possible links between the elements which is in $O(n^2)$. Hence we provide a method that allows to extract k clusters on request from the quasi-dendrogram.

Exactly like with a dendrogram, if there is only one level with k clusters it is simple as we just return this level of the quasi-dendrogram. However it is possible that several levels have the same number of clusters and in this case we have to determine which one covers the best the data. We base our solution on the notion of *persistence* (Definition 15) closely related to the notion of stability mentioned by Campello et al. 2015.

Definition 15 (Persistence)**.** *We define the persistence of a cluster C in the hierarchy h as:*

$$persistence(C, h) = d_{death}(C, h) - d_{birth}(C, h)$$

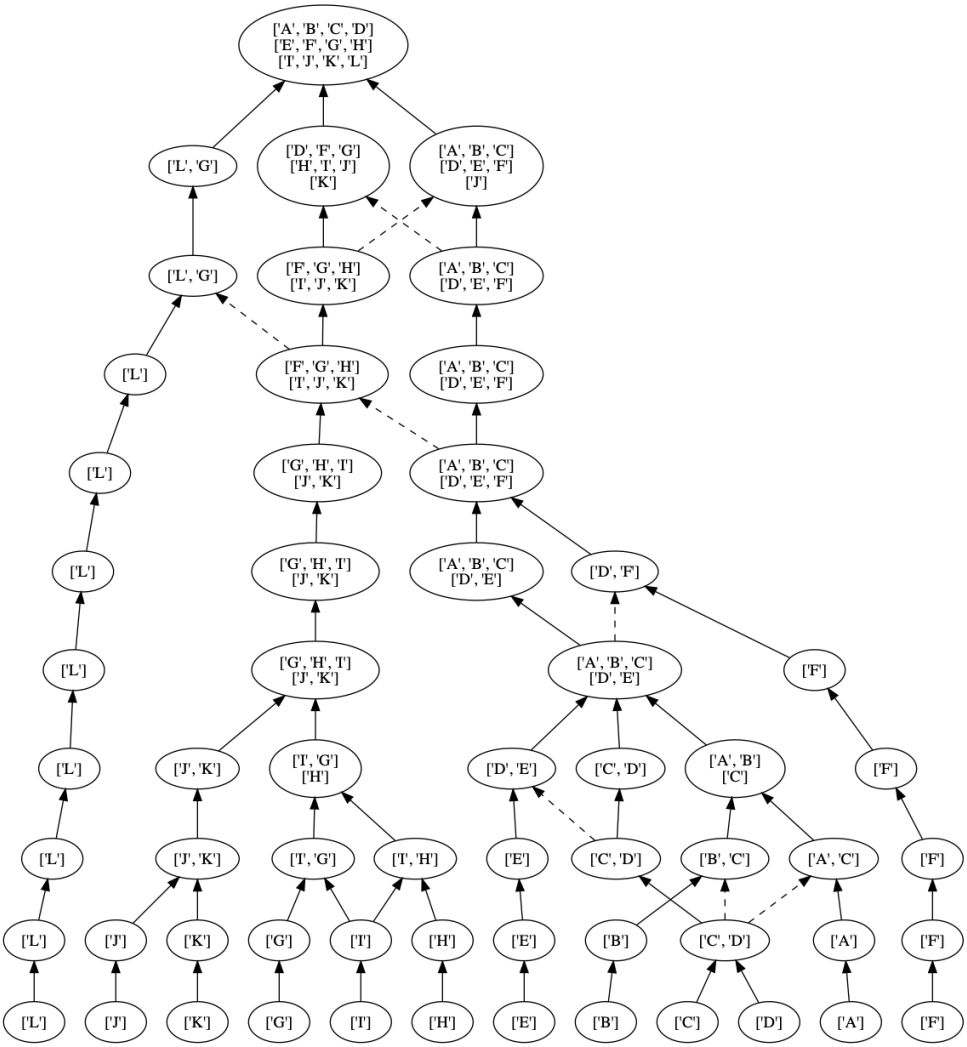


Figure 2.4: *OHC* quasi-dendrograms obtained from the hand-built example in Figure 2.3a for a merging criterion of 0.2.

where d_{birth} is the distance from which the cluster C appears in the hierarchy and d_{death} is the distance from which the cluster C disappears from the hierarchy by being merged with another cluster. We have $0 \leq d_{birth} < d_{death} \leq 1$ so the persistence represents the fraction of existence of the cluster in the hierarchy.

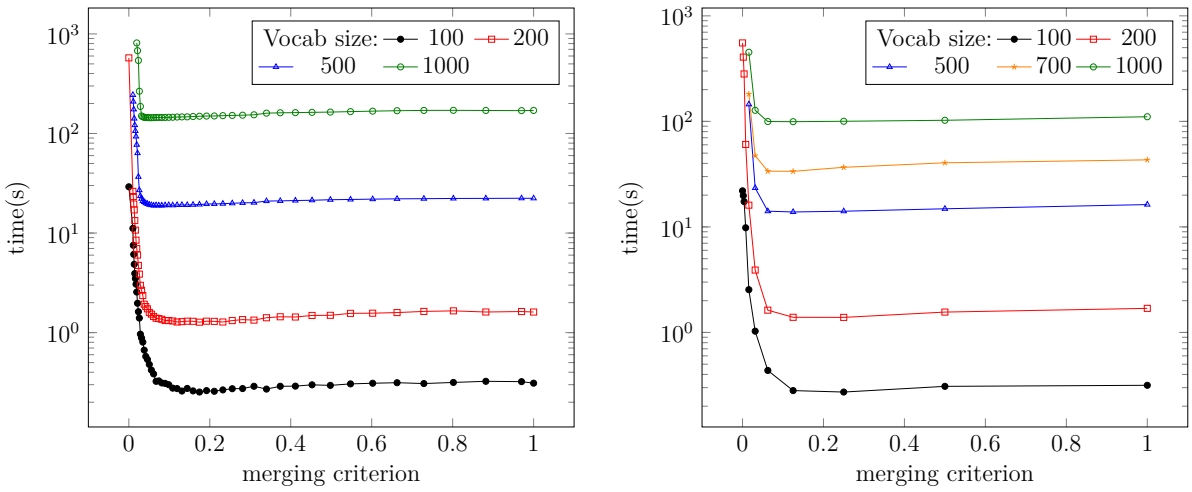
The persistence of a level l of the hierarchy h is then given by:

$$persistence(l, h) = mean\{persistence(C, h) \mid C \in l\}$$

A cluster that appears only briefly in the hierarchy is not stable and seems less interesting as it is an intermediate to more persistent clusters. Hence to select a cover of k clusters among several possible levels we compute the persistence of all of them and chose the level with the highest value. This way the selected cover will have the most robust clusters, those that lasted the longest.

2.6.4 Time complexity of the OHC algorithm

Study of the merging criterion



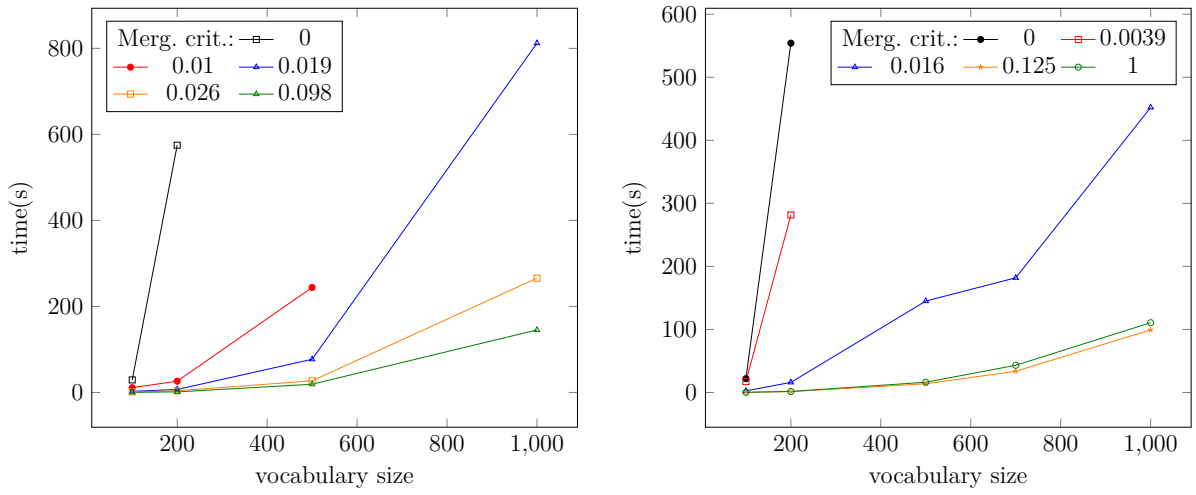
(a) Wiley dataset for different vocabulary sizes. (b) dblp dataset for different vocabulary sizes.

Figure 2.5: Execution time according to the merging criterion on different datasets.

We observe in Figure 2.5 for all the datasets that when the merging criterion increases the execution time decreases significantly even in semi-logarithmic scale on the y-axis.

We can also notice a slight increase in the time taken after reaching a merging criterion of around 0.2. It is due to the fact that when the merging criterion increases we are more likely to completely merge clusters so we reach faster the top of the hierarchy. It means less levels and less overlapping clusters so less computation. However in this case we have the same drawback of chaining effect as the single-linkage clustering that we wanted to avoid.

Study of the size of the vocabulary



(a) Wiley dataset for different merging criteria. (b) dblp dataset for different merging criteria.

Figure 2.6: Execution time according to the size of the vocabulary on different datasets.

Similarly and as expected we observe in Figure 2.6 that when the size of the vocabulary increases the execution time increases significantly. However depending on the merging criterion the complexity is not the same. The complexity seems exponential for a merging criterion equal to 0 and quadratic for a merging criterion of 1. Between these extreme values the complexity is polynomial with a degree increasing when the merging criterion moves from 1 to 0. Raw data of these graphs can be found as double entry tables in Appendix 2.8.

The trends are even clearer in Figure 2.7 where we performed the OHC algorithm on random 2-dimensional vectors. For a merging criterion greater than 0.01 (the blue line and below), when the number of vectors increases, the first limitation becomes the pairwise distance computation used as input of every agglomerative clustering algorithm. On the

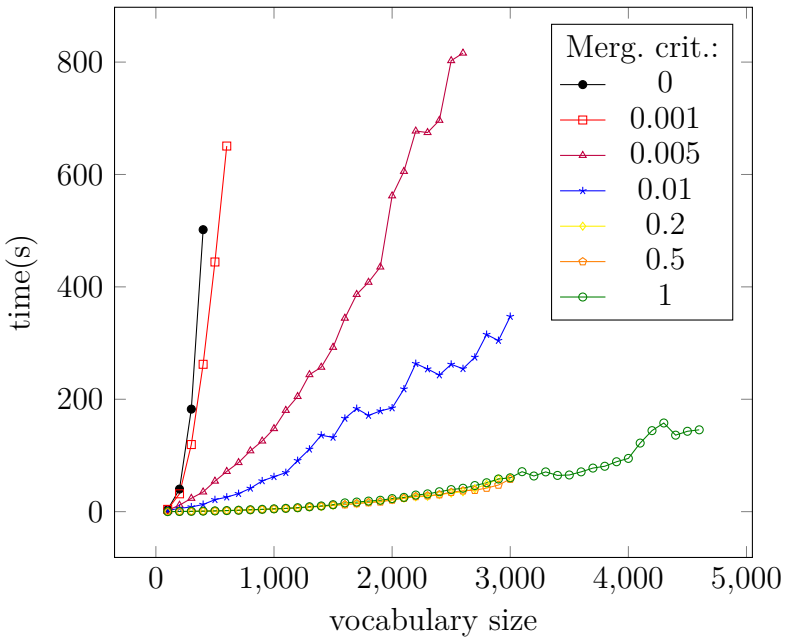


Figure 2.7: Execution time according to the size of the vocabulary for random generated 2-dimensional vectors.

contrary, above the blue line, the limitation is the execution of the OHC algorithm by itself compared to the pairwise distance computation of the vectors.

2.6.5 Optimisation

Some simple techniques were used to slightly optimize the OHC algorithm. It consists in keeping in memory time consuming intermediate results computed at the step n to reuse them at the step $n+1$. It is very similar to a dynamic programming technique. For instance for each current cluster we preserve and maintain the list of adjacent vertices and their degree of adjacency. The degree of adjacency being the number of edges that connects an adjacent vertex to the vertices of a cluster is computed in linear time according to the size of the cluster which becomes time consuming at the end of the process when the clusters are big.

To significantly speed up the process and scale up our algorithm it is also possible to precompute a set of possibly overlapping clusters over a given δ-neighbourhood graph with a classical method, for instance *CLIQUE* (Agrawal et al. 2005), and build the OHC hierarchy on top of that. This is similar to techniques like *EAGLE* (Shen et al. 2009) but building a quasi-dendrogram instead of the classical tree.

2.7 Conclusion

In this Chapter we proposed an overlapping hierarchical clustering (OHC) framework. We construct a quasi-dendrogram hierarchical structure to represent the clusters that is however not necessarily a tree (of specific shape) but a directed acyclic graph. In this way, at each level, we represent a set of possibly overlapping clusters. If the clusters present in the data show no overlaps, the obtained clusters are identical to the clusters we can compute using agglomerative clustering methods. In case of overlapping and nested clusters, however, our method results in a richer representation that can contain relevant information about the structure of the clusters of the underlying dataset.

2.8 Detailed figures and tables related to the OHC algorithm

Quasi-dendrograms

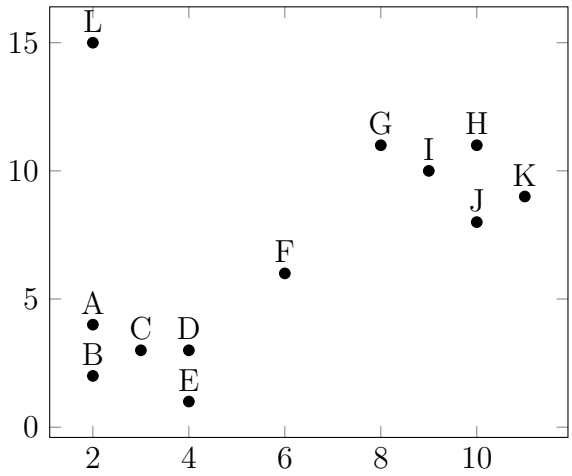


Figure 2.8: An example of 12 two-dimensional points to cluster.

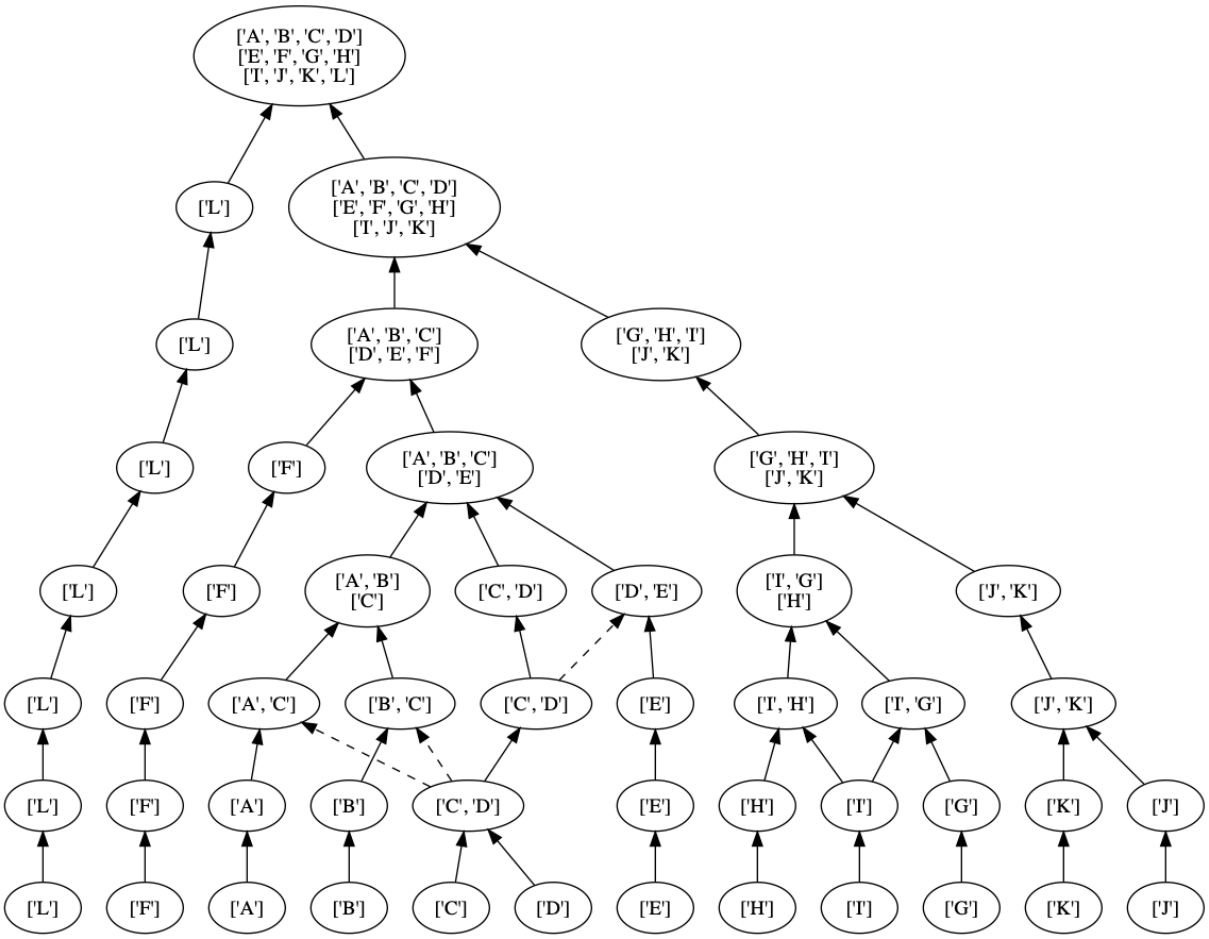


Figure 2.9: *OHC* quasi-dendrograms obtained from the hand-built example in Figure 2.8 for a merging criterion of 0.3.

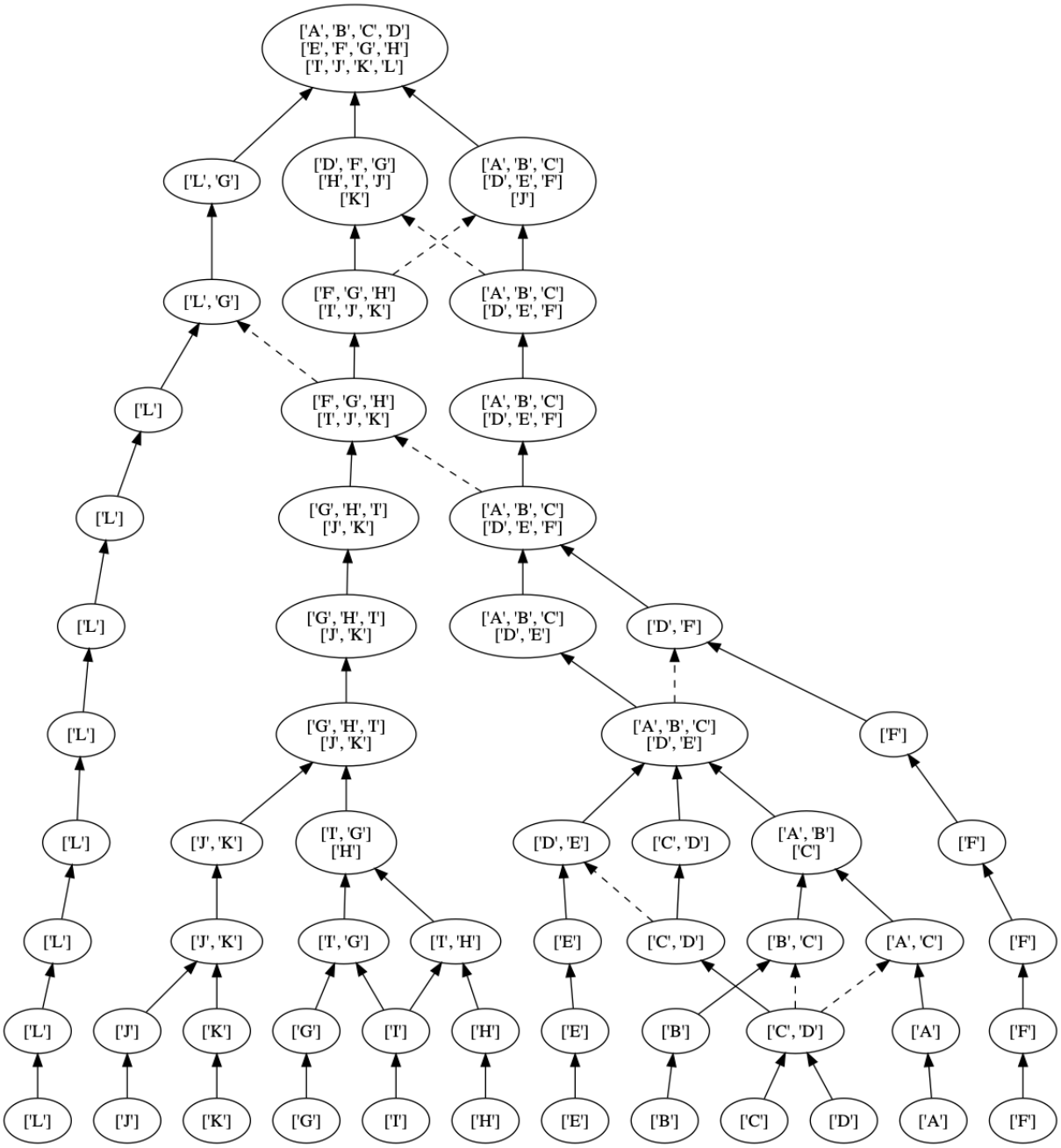


Figure 2.10: *OHC* quasi-dendrograms obtained from the hand-built example in Figure 2.8 for a merging criterion of 0.2.

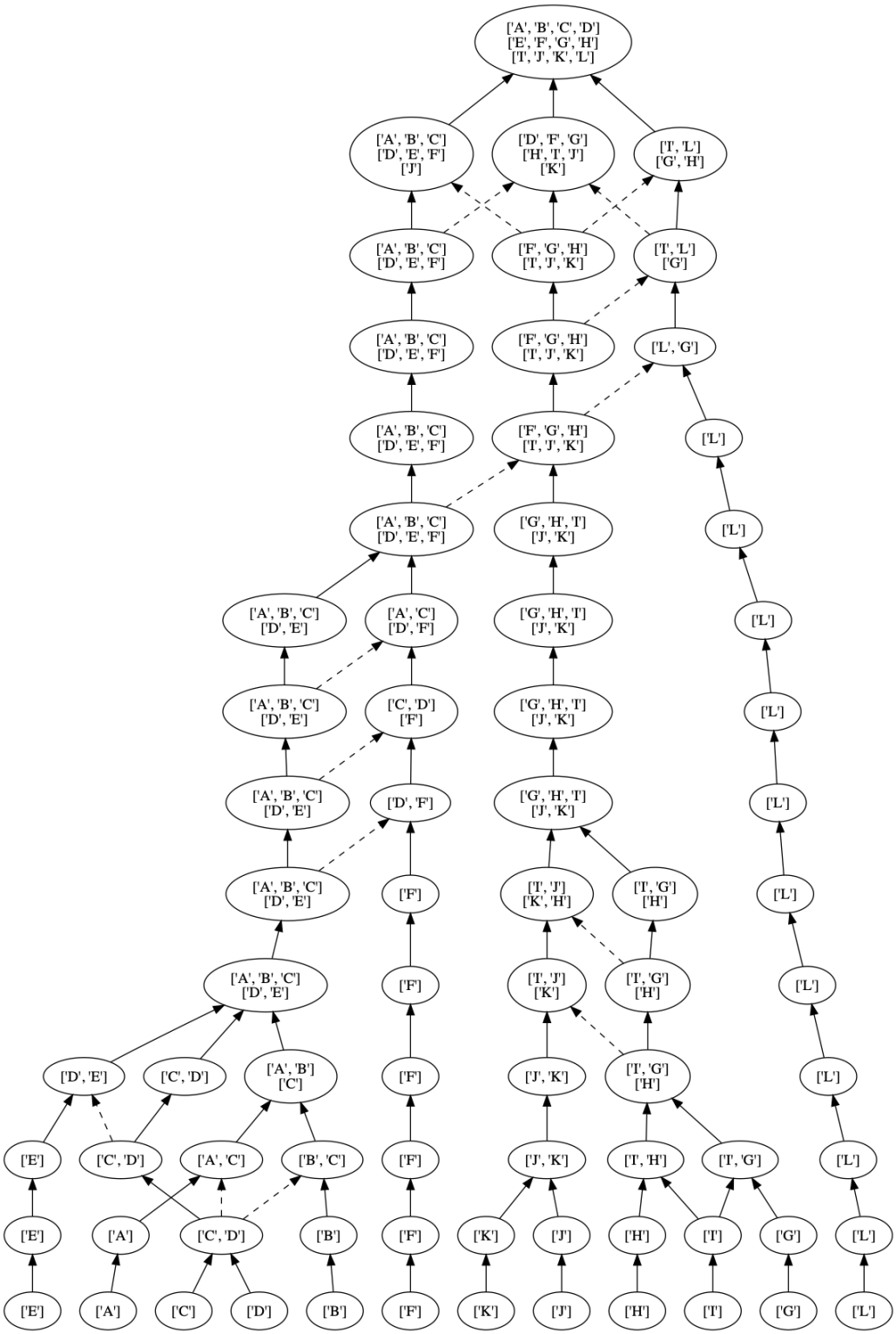


Figure 2.11: *OHC* quasi-dendrograms obtained from the hand-built example in Figure 2.8 for a merging criterion of 0.1.

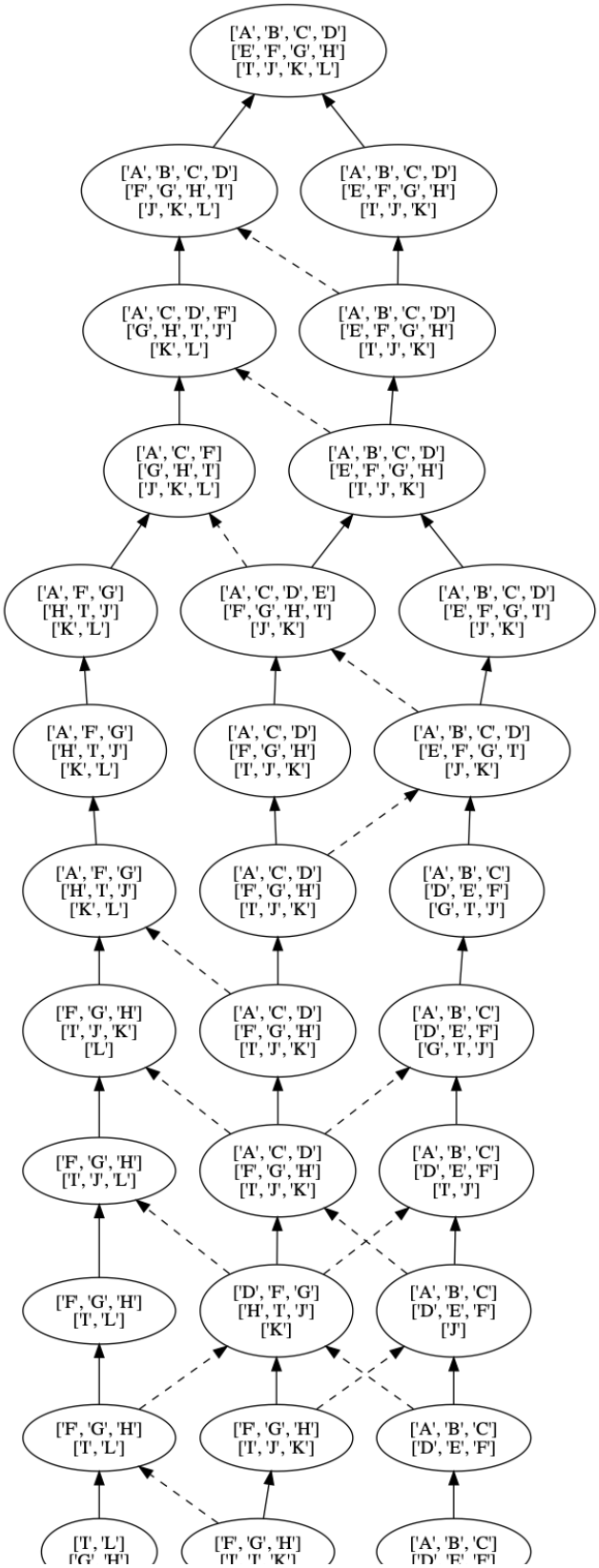


Figure 2.12: *OHC* quasi-dendrograms obtained from the hand-built example in Figure 2.8 for a merging criterion of 0. Upper part.

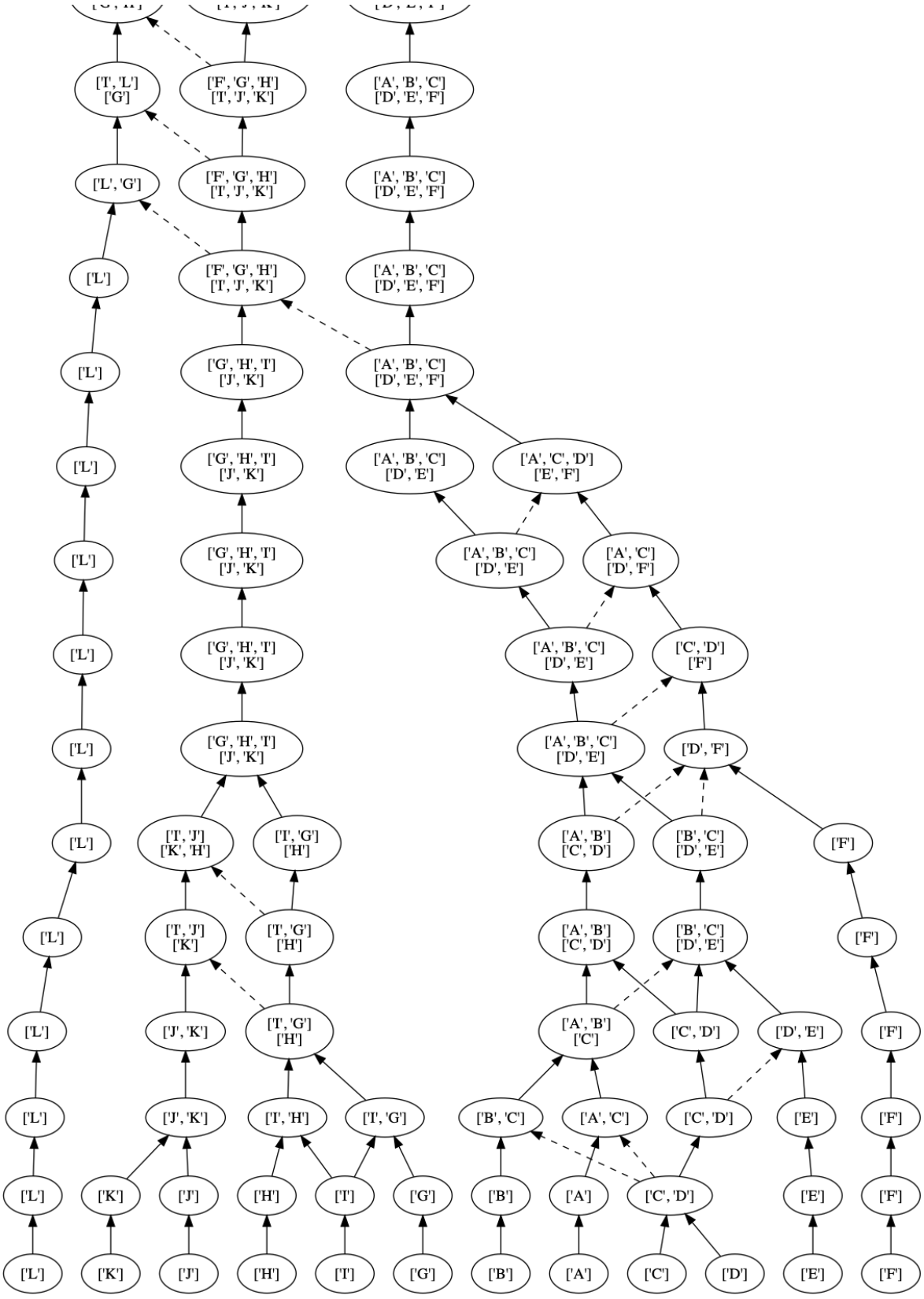


Figure 2.13: *OHC* quasi-dendrograms obtained from the hand-built example in Figure 2.8 for a merging criterion of 0. Lower part.

Execution time

merging criterion	100	200	500	1000
0	29.15	574.75		
$1 \cdot 10^{-2}$	11.15	26.23	244.12	
$1.1 \cdot 10^{-2}$	7.54	23.27	209.23	
$1.21 \cdot 10^{-2}$	6.11	19.59	174.69	
$1.33 \cdot 10^{-2}$	4.86	17.08	141.27	
$1.46 \cdot 10^{-2}$	3.92	13.35	121.43	
$1.61 \cdot 10^{-2}$	3.49	10.7	106.16	
$1.77 \cdot 10^{-2}$	3.07	8.47	93.32	
$1.95 \cdot 10^{-2}$	2.56	6.98	77.13	812.04
$2.14 \cdot 10^{-2}$	1.97	6.01	63.53	678.99
$2.36 \cdot 10^{-2}$	1.62	4.73	36.77	541.4
$2.59 \cdot 10^{-2}$	1.4	3.86	26.98	265.72
$2.85 \cdot 10^{-2}$	0.97	2.99	23.57	186.94
$3.14 \cdot 10^{-2}$	0.88	2.66	22.14	149.84
$3.45 \cdot 10^{-2}$	0.8	2.36	21	147.2
$3.8 \cdot 10^{-2}$	0.67	1.93	20.53	145.1
$4.18 \cdot 10^{-2}$	0.58	1.83	19.97	144.83
$4.59 \cdot 10^{-2}$	0.54	1.73	19.67	144.81
$5.05 \cdot 10^{-2}$	0.48	1.59	19.28	144.02
$5.56 \cdot 10^{-2}$	0.42	1.53	19.09	144.11
$6.12 \cdot 10^{-2}$	0.38	1.45	18.89	144.29
$6.73 \cdot 10^{-2}$	0.32	1.4	19.02	144.79
$7.4 \cdot 10^{-2}$	0.33	1.38	18.92	144.97
$8.14 \cdot 10^{-2}$	0.31	1.36	18.93	144.81
$8.95 \cdot 10^{-2}$	0.31	1.32	19.01	144.96
$9.85 \cdot 10^{-2}$	0.3	1.32	19.11	145.34
0.11	0.28	1.31	19.01	145.65
0.12	0.27	1.28	19.13	146.37
0.13	0.26	1.29	19.16	146.76
0.14	0.28	1.3	19.13	146.94
0.16	0.26	1.3	19.25	148
0.17	0.25	1.27	19.36	149

merging criterion	100	200	500	1000
0.19	0.26	1.3	19.57	149.67
0.21	0.26	1.3	19.63	149.96
0.23	0.27	1.28	19.73	151.03
0.26	0.27	1.32	19.91	151.91
0.28	0.27	1.35	20.1	152.71
0.31	0.29	1.34	20.21	154.06
0.34	0.27	1.41	20.96	160.03
0.37	0.29	1.44	21.04	161.37
0.41	0.29	1.43	21.22	161.98
0.45	0.3	1.49	21.35	163.12
0.5	0.3	1.49	21.62	164.05
0.55	0.31	1.56	21.75	165.84
0.6	0.31	1.57	21.91	167.64
0.66	0.31	1.59	22.07	169.34
0.73	0.31	1.63	22.14	170.56
0.8	0.32	1.65	22.26	170.81
0.88	0.32	1.61	22.29	170.27
0.97	0.32	1.63	22.36	169.66
1	0.31	1.61	22.33	170.39

Table 2.2: Execution time (s) of the OHC algorithm depending on the merging criterion for different vocabulary sizes on the **Wiley** dataset.

merging criterion	100	200	500	700	1000
1	0.32	1.69	16.28	43.2	110.66
0.5	0.31	1.56	14.86	40.5	102.43
0.25	0.27	1.39	14.08	36.71	100.24
0.13	0.28	1.39	13.86	33.63	99.36
$6.25 \cdot 10^{-2}$	0.44	1.63	14.1	33.74	99.72
$3.13 \cdot 10^{-2}$	1.03	3.9	23.42	47.36	127.34
$1.56 \cdot 10^{-2}$	2.54	16.08	145.04	181.94	452.14
$7.81 \cdot 10^{-3}$	9.82	60.45			
$3.91 \cdot 10^{-3}$	17.38	281.59			
$1.95 \cdot 10^{-3}$	19.78	406.42			
0	21.98	554.06			

Table 2.3: Execution time (s) of the OHC algorithm depending on the merging criterion for different vocabulary sizes on the **dblp** dataset.

Chapter 3

Hierarchical structure similarity

3.1 Introduction

We presented in the previous Chapter a new method of hierarchy construction over a set of term vectors and explored the expressiveness of the produced structures, the quasi-dendrograms, and how they differ from the classical dendrograms by construction and visually. However the quasi-dendrogram can be a very complex structure making it difficult to explore. Also for our specific task where we try to build a hierarchy of scientific domains, there is no ground truth on how this structure should be. It strongly depends on the data we use and the period we study. Then having a proper method to compare different structures appears essential. The goal is not to tell how good a hierarchical structure is but how it differs from another one. In this Chapter we present a new similarity measure that can compare both dendrograms and quasi-dendrograms.

3.2 Definition of the similarity measure

As there is no ground truth on the hierarchy of the data we used, we need a similarity measure to compare the hierarchical structures produced by hierarchical clustering algorithms. The goal is not only to compare the topology but also the content of the nodes of the structure. However up to our knowledge there is very little in the literature about hierarchy comparison especially when the structure is similar to a DAG or a quasi-dendrogram. Fowlkes and Mallows 1983 defined a similarity measure per level and the new similarity function we propose is based on the same principle. First we construct a similarity between two given levels of the hierarchies, and then we extend it to the global structures by exploring all the existing levels.

3.2.1 Level similarity

Given two hierarchies h_1 and h_2 and a cardinality i, we assume that it is possible to identify a set l_1 (resp. l_2) of i clusters for a given level of hierarchy h_1 (resp. h_2). Then, to measure the similarity between l_1 and l_2, we take the maximal Jaccard similarity among one cluster of l_1 and every clusters of l_2. The average of these similarities, one for each cluster of l_1, will give us the similarity between the two sets. If we consider the similarity matrix of h_1 and h_2 with a cluster of l_1 for each row, a cluster of l_2 for each column and the Jaccard similarity between each pair of clusters at the respective coordinates in the matrix, we can compute the similarity between l_1 and l_2 by taking the average of the maximal value for each row. Hence, the similarity function between two sets of clusters l_1, l_2 is defined as:

$$sim_l(l_1, l_2) = mean\{max\{J(c_1, c_2) \mid c_2 \in l_2\} | c_1 \in l_1\} \tag{3.1}$$

where J is the Jaccard similarity function (see Section 1.2.5).

However, taking the maximal value of each row shows how the clusters of the first set are represented in the second. If we take the maximal value of each column we will see the opposite, i.e. how the second set is represented in the first set. Hence with this definition the similarity might not be symmetrical so we propose this corrected similarity measure that shows how both sets are represented in the other one:

$$sim_l^*(l_1, l_2) = mean(sim_l(l_1, l_2), sim_l(l_2, l_1)) \tag{3.2}$$

3.2.2 Global similarity

Now that we can compare two levels of the hierarchical structures, we can simply average the similarity for each corresponding levels of the same size. For classical dendrograms, each level has a distinct number of clusters so identification of levels is easy. Conversely, our quasi-dendrograms may have several distinct levels (pseudo-levels) with the same number of clusters. If so, we need to find the best similarity between these pseudo-levels. For a given level (i.e. number of clusters), we want to build a matching M that maps each pseudo-level $l_1^1, l_1^2, ...$ of h_1 to at least one pseudo-level $l_2^1, l_2^2, ...$ of h_2 and conversely (see Figure 3.1). This matching M should maximize the similarity between pseudo-levels while preserving their hierarchical relationship. That is, for a, b, c, d representing the height of pseudo-levels in the hierarchies, if $(l_1^a, l_2^c) \in M$ and $(l_1^b, l_2^d) \in M$, then

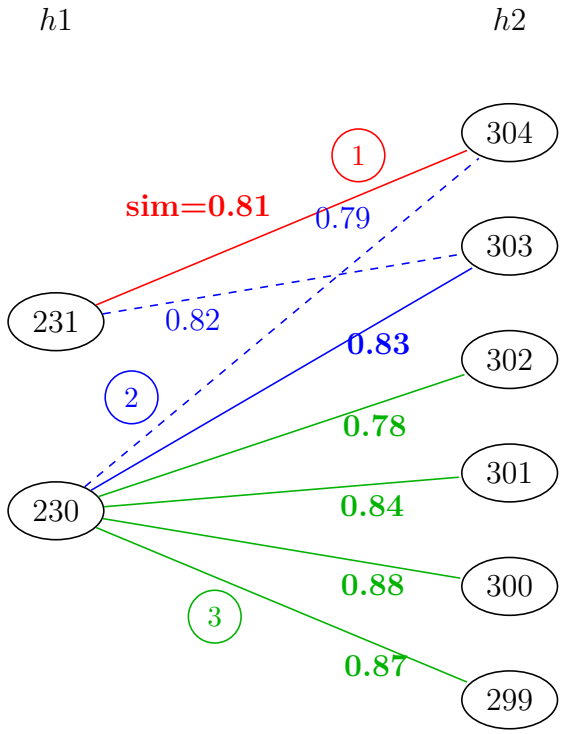


Figure 3.1: Computing the similarity between two quasi-dendrograms h_1 and h_2 for levels having the same number of clusters.

$(b \geq a \rightarrow d \geq c)$ or $(b < a \rightarrow d < c)$. The hierarchy prevents any "crossings" in M, such as (l_1^{231}, l_2^{303}) with (l_1^{230}, l_2^{304}).

Considering the subsets of pseudo-levels of each hierarchy, to produce this mapping, our simple algorithm is the following.

— (1): We initialize M and two pointers with the two highest pseudo-levels. They are the first math to add to M. (l_1^{231}, l_2^{304}) is added to M in our example in Figure 3.1.

— (2): Then at each step, for each hierarchy, we consider current pointers and their child pseudo-level, and compute their pairwise similarities. We add pseudo-levels with maximal similarity to M. From the three possibilities, (l_1^{230}, l_2^{303}) presents the maximal similarity in Figure 3.1. Whenever a child is chosen, the respective pointer advances, and at each step, at least one pointer advances.

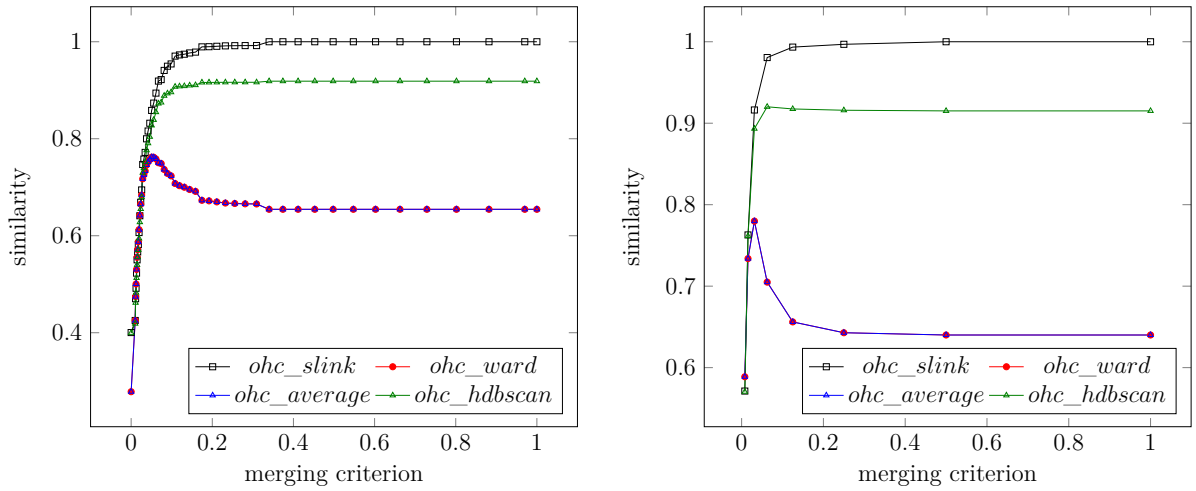
— (3): Once pseudo-levels have been consumed on one side, ending with l^f, we can finish the process by adding (l^f, l') to M for all remaining pseudo-level l' on the other side. $l^f = l_1^{230}$ in our example and the final matching is $M = \{(l_1^{231}, l_2^{304}), (l_1^{230}, l_2^{303}), (l_1^{230}, l_2^{302}), (l_1^{230}, l_2^{301}), (l_1^{230}, l_2^{300}), (l_1^{230}, l_2^{299})\}$.

Finally, from Equation 3.2 we define the similarity between two hierarchies as

$$sim(h_1, h_2) = mean\{sim_l^*(l_1, l_2) | (l_1, l_2) \in (h_1, h_2) \ \& \ (l_1, l_2) \in M\}. \tag{3.3}$$

3.3 Comparison of hierarchical structures

This similarity measure allows us to compare the quasi-dendrograms between themselves but above all quasi-dendrograms produced by the OHC algorithm to classical HAC algorithms. The goal is to show how the quasi-dendrograms evolve according to the merging criterion with regard to a given baseline. We recall that this baseline is not the ground truth, only a point of comparison. We chose the baseline been the classical single-linkage algorithm and in particular its *SLINK* implementation (Sibson 1973). Its will allow to illustrate by the experiment the Property 3. We will also compare the quasi-dendrograms to other classical algorithms such as *HDBSCAN** (Campello et al. 2015) and the *Ward* method of (Ward Jr 1963). Details on these algorithms are given in Section 1.2. The datasets are the hierarchies produced using the OHC algorithm based on the datasets used in the previous Chapter (Section 2.6.1).



(a) Wiley dataset for different hierarchical algorithms.

(b) dblp dataset for different hierarchical algorithms.

Figure 3.2: Similarity with classical algorithms according to the merging criterion on different datasets.

As we can see in Figure 3.2 for both datasets, when the merging criterion increases we obtain a hierarchy more and more similar to the one produced by the classical *SLINK*

algorithm until we obtain exactly the same for a merging criterion of 1. Knowing this fact it is also normal to have a similarity between *OHC* and the other algorithms converging to the similarity between *SLINK* and these algorithms. If compared to *HDBSCAN** the similarity has the same behaviour than compared to *SLINK*, we can notice that *OHC* and *Ward* hierarchies are the most similar for a merging criterion smaller than 1, around 0.05. The same is true between *OHC* and *average-linkage* hierarchies as *Ward* and *average-linkage* clustering give very similar hierarchies.

3.4 Analysis and possible variants

3.4.1 Detailed similarity

The similarity measure described above gives an average similarity between corresponding level of the hierarchies. It is interesting to have a global value telling us how close are two quasi-dendrograms. However it does not allow us to detect the parts of the hierarchies that differ. For a better understanding of this changes and similarities, we offer the possibility to access to a detailed version of the similarity measure where we return the list of matching levels and their associated similarity.

The similarity is not computed branch by branch but level by level. Hence this detailed view will not allow us to detect common branches, i.e. common sub-hierarchies on the structure of science, but it will give us information on how abstract topics from higher levels of the hierarchies are similar compared to topics from lower levels.

On Figure 3.3 that represents the detailed similarity between two quasi-dendrograms computed on different periods of the Wiley dataset, the levels 0 correspond to the bottom levels of the hierarchies when the levels 125 and 115 correspond to the respective roots of the first and second quasi-dendrograms. The trace left by the similarity function appears in colors that represent the similarity between matching levels. The color scale on the right shows a greater similarity in yellow to a lower similarity in purple. We need to note that it is not a 0 to 1 scale and even a low similarity still means a level similarity of 0.65. We can see on this Figure that the upper halves of the hierarchies are similar in term of structure as the path taken by the similarity function stays on the diagonal with only slight deviations while the lower halves differ more with bigger deviations. On the contrary, the clusters seem to be less similar in the upper part that appears more purple than in the lower part.

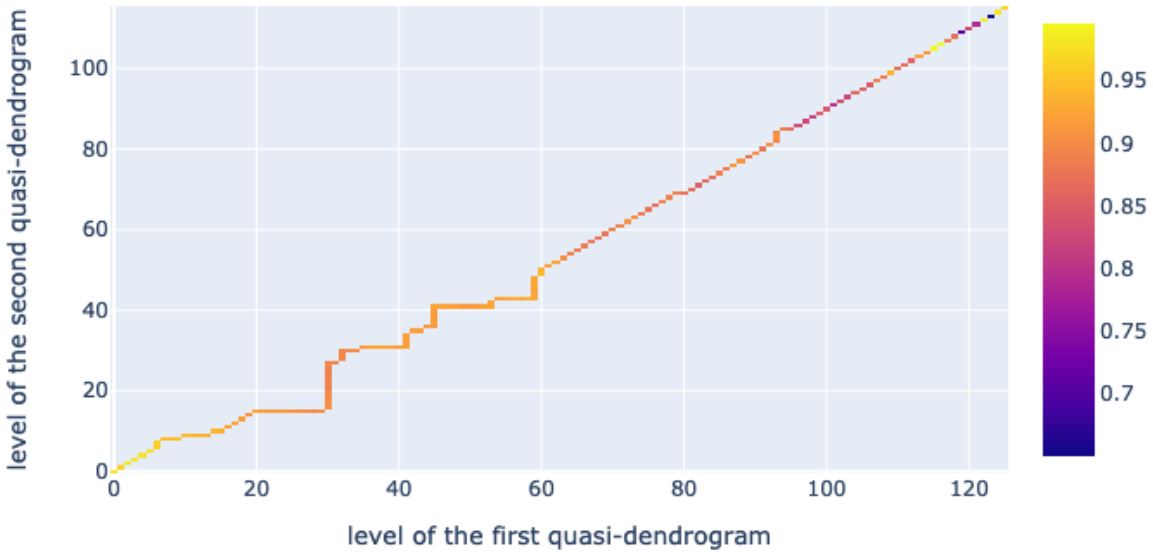


Figure 3.3: Evolution of the matching levels during the similarity computation for 2 hierarchies from the Wiley dataset.

3.4.2 Average versus weighted similarity

Also the level similarity defined in 3.2 is based on a simple average similarity between corresponding clusters. This gives the same importance to all the clusters, regardless of their size. It implies that a single noise term that persists in the quasi-dendrogram will have the same weight for the similarity computation than a big abstract cluster that forms a relevant scientific domain. To reduce this impact and reflect more the reality of the cluster distribution we proposed a variant of the level similarity that adds a weight to the similarity associated to each cluster by multiplying it by a coefficient corresponding to the proportion of the terms contained in the cluster over the sum of the terms of all the clusters of the level. This can be formulated as follow:

$$weighted\_sim_l(l_1, l_2) = \frac{\sum_{c_1 \in l_1} (|c_1| * max\{J(c_1, c_2) \mid c_2 \in l_2\})}{\sum_{c_1 \in l_1} |c_1|} \tag{3.4}$$

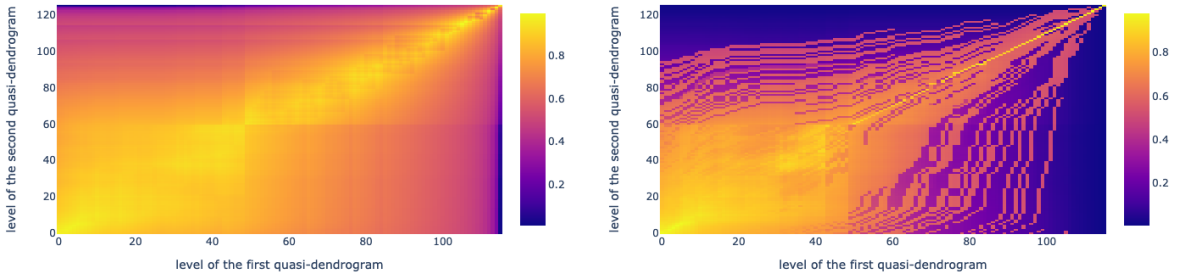
where J is the Jaccard similarity function.

It is important to notice that in the denominator as the clusters can overlap, $\sum_{c_1 \in l_1} |c_1|$ is not equivalent to $|\cap_{c_1 \in l_1} c_1|$. It means that a term that appear in several clusters will be counted with its multiplicity in the denominator.

Similarly to the Equation 3.2, a corrected version of the weighted similarity is given by:

$$weighted\_sim^*_l(l_1, l_2) = mean(weighted\_sim_l(l_1, l_2), weighted\_sim_l(l_2, l_1)) \tag{3.5}$$

Figure 3.4 presents the heatmap of level similarity between each level of two quasi-dendrograms. Figure 3.4a shows the pairwise similarity for the regular similarity function while Figure 3.4b shows the pairwise similarity for the weighted variant. The color scale is similar for both Figures and we can clearly see that when bigger clusters appear in the upper part of the hierarchies, the weight given to the cluster significantly changes the similarity value. Only the bottom left quarter is relatively similar for weighted and non-weighted variants which correspond to a pairwise similarity between low levels of both quasi-dendrograms.



(a) Average pairwise level similarity.

(b) Weighted pairwise level similarity.

Figure 3.4: Study of the similarity function between 2 OHC hierarchies based on the Wiley dataset.

3.5 Similarity as a search algorithm

Mapping the pairs of matched levels on a heatmap like in Figure 3.3 makes the similarity function looking like a *pathfinding algorithm*. Indeed, considering the way the similarity is computed, from the top of the hierarchies to the bottom, it acts like if the objective was to determine the shortest path from the top right corner of the heatmap, that is the top of both hierarchies, to the bottom left corner, corresponding also to the bottom of the hierarchies. The cost of the path being the sum of level similarities it is composed.

More precisely it mimics the behaviour of a *greedy algorithm* when several levels with the same number of clusters appear in both hierarchies. Under the constraint that it is not possible to turn back and move up in the quasi-dendrograms, at each step there are only three possible moves to reach the bottom. A down or left move that corresponds to a move in only one of the hierarchies or a diagonal move that combine the two previous moves. There is not anticipation and a heuristic function that would indicate which move to chose is simply the value of similarity between levels of the destination. There is no guaranty that this greedy algorithm will find an optimal solution as it looks only for local optimum but when the depth of the hierarchies increases it becomes impossible to computed all the pairwise level similarities. Hence the goal here was only to approximate a globally optimal solution in a reasonable amount of time.

3.6 Conclusion

In this Chapter we explored the possibility to compute a quasi-dendrogram similarity that allow to compared these structures with also classical dendrograms. To our knowledge, very little exists in the literature when the structure becomes more complex than a simple tree. Hence we proposed a new similarity measure that approaches the problem with a level-wise point of view. Indeed, if for two hierarchical structures, when we extract a k-cover we obtain similar clusters for any value of k, it means that the structures are very similar and conversely. We used this similarity measure to confirm by experiments that the OHC algorithm, defined in the previous Chapter, acts like a classical single-linkage algorithm when the merging criterion is close to 1 and differs more and more when this value decreases. We also proposed a detailed view of the similarity to detect common and different parts of the hierarchies and a weighted variant with the goal to reduce the impact caused by small noise clusters that would persist in the hierarchies. In fact detecting similar parts across several hierarchies, i.e. that are produced for several merging criterion, can be seen as a robustness criterion. This could be explored and combined to the notion of persistence to determine the most robust clusters of the quasi-dendrograms. Finally we showed the relationship between this similarity measure and a potential pathfinding algorithm which could be a very interesting axis of research for a future work.

Chapter 4

Temporal analysis of hierarchical structures

4.1 Introduction

Back to the main objective of this thesis, the goal is to offer methods to social scientists that will help them to analyze and better understand the intrinsic structure of science and the processes of evolution that are at stake. The previous Chapters provided a method to extract a complex hierarchical structure of key terms from a set of scientific papers and a similarity measure to compare diverse types of hierarchical structures. So far we were only interested in determining the structure of science for a given period of time without considering any evolution. This is the subject of this Chapter where we present a method to build an evolutionary map based on a set of hierarchical structures.

4.2 Objectives and framework

Reconstructing the maps of the evolution of the species is a central question in biology. It leads to the production of the well known phylogenetic trees that separates the species by their last common ancestor. While the term *evolution* has a very specific meaning in biology, one often refers to the evolution of other objects when the object undergoes small gradual changes. For example, scientific concepts often gradually change their meaning or get connected to other concepts. It is not an easy task to understand these gradual changes since the scientific literature is rather large and overspecialized.

Such structures that mimic the gene evolution for scientific domains take various names and shapes in the literature, several are detailed in Section 1.3. Some focus on the study of the relation between the publication by looking at co-citation networks (C. Chen 2004, Rosvall and Bergstrom 2010) while others focus on the content of the papers (Cui et al. 2011, Shahaf, Yang, et al. 2013, Chavalarias and Cointet 2013). If the co-citation study can

only give a vision at the scale of a document, textual analysis allows a thinner granularity by showing the evolution of sets of words. Our task consisting in analysing the text of scientific papers will naturally guide us to the evolution of cluster of terms. Evolutionary maps of term clusters are called alternatively TextFlow (Cui et al. 2011), metro maps (Shahaf, Yang, et al. 2013) or phylomemetic structures (Chavalarias and Cointet 2013).

Input structures The Overlapping Hierarchical Clustering (OHC) algorithm (Section 2.4) detects dense areas of an embedding space and groups their keywords to form scientific topics and that in a hierarchical fashion. It produces structures called quasi-dendrograms (Section 2.3.4) that can be seen as specific semi-lattices.

As seen in Section 2.6.1 we can produce a set of hierarchies based on scientific papers from Wiley or dblp by grouping the data by periods of 3 years with a sliding window of 1 year. This sliding window is here to attenuate the changes that can occur in the hierarchical structures. Indeed having a part of the data in common will increase the probability of having common terms in the vocabularies. As their representation is given by the documents in which they occur, these common keyterms will also have similar relationship consequently making the hierarchies similar too.

Also, in such hierarchy the notion of level, related to a distance graph, is important and similar bags of words at different levels are considered separately. If it is interesting for studying the structure of science of a given period of time, there is no particular reason to keep this separation when we want to follow the evolution of these topics. Hence we simplify this structure into a directed acyclic graph (DAG) that contains strictly different bags of words from the quasi-dendrogram as nodes but by preserving the parent relationship of the topics.

Framework In our case, following the evolution of one given topic through time is easy as one can simply look for the most similar bag of words in hierarchies from consecutive time periods. The topic similarity is seen here in the sense of the Jaccard similarity, i.e. the ratio of the intersection of bags of words over their union. This is on what most of the evolutionary structures are based. However doing this we completely put aside the notion of structure given by the hierarchies and if we want to follow the evolution of several topics we will lose their relationship. One way to keep that is to align the hierarchies and then follow the alignments to have the evolution of the topics.

This will constitute our two constraints to align the hierarchies, first we want to align

the topics, i.e. the **content of the nodes** of the hierarchies, so that the following topics will make sens through the different periods. And as a second constraint we also want to preserve the **structure of the hierarchies**, meaning that the topology of the hierarchies should be taken into account during the alignment.

Also it is reasonable to say that following the evolution of similar structures will lead to better results. Similar meaning having an equivalent size which corresponds to the depth and the width of the hierarchical structure. These two values are correlated to specific parameters that we should set to the same value to have structures of similar size. The width corresponds directly to the number of leaves of the quasi-dendrogram, that is to the size of the vocabulary on which the hierarchy is produced. The depth is correlated to the value of the merging criterion used to generate the hierarchy. It is not a direct causality as we cannot determine the depth of the quasi-dendrogram from the merging criterion but having a "permissive" merging criterion, i.e. close to 1, will generate a shallow hierarchy as it is easier to merge clusters. At the opposite, having a merging criterion close to 0 will lead to very deep structures. Hence we will follow the evolution of scientific topics from a set of quasi-dendrograms based on the same number of key terms and produced with the same merging criterion.

4.3 Evolutionary map

The concept of evolutionary map is quite simple and is defined in Definition 16. It is similar to structures like the *metro map* from Shahaf, Yang, et al. 2013 and the *phylomemetic structure* from Chavalarias and Cointet 2013.

Definition 16 (Evolutionary map)**.** *An evolutionary map EM can be seen as a directed acyclic graph made of a set of timelines.*

- ***Temporal node**: A node of the evolutionary map is a set of words that represents a topic of a given period of time.*
- ***Timeline**: A timeline connects a series of nodes of consecutive periods of time.*

Like in other evolutionary structures, interesting points of interest can be found in our evolutionary map. Each timeline starts with an **emerging node** that has no ancestor and finishes with a **declining node** that as no successor. In between, merging and splitting nodes can occur. A **merging node** appears when two timelines share a common node, i.e the same set of terms for the same period, but with different predecessors. In the same way

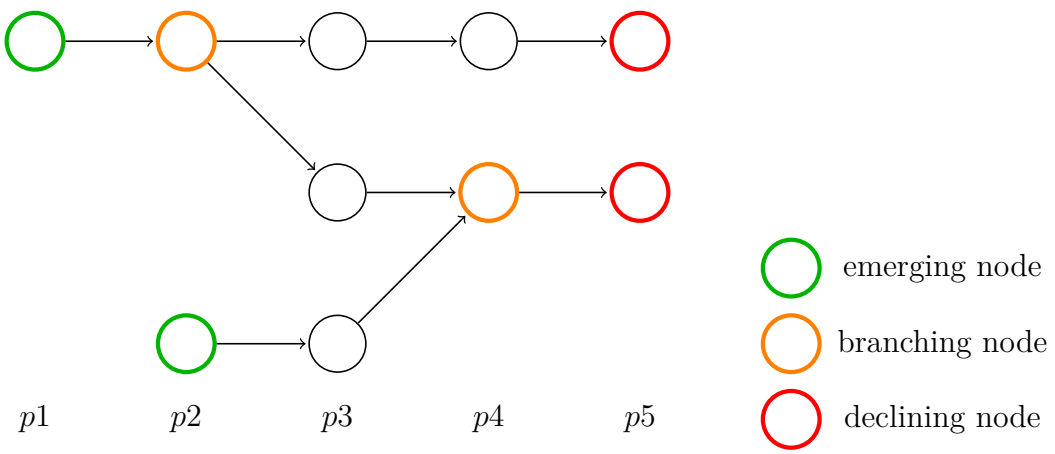


Figure 4.1: Special events occurring in the evolutionary map.

a **splitting node** appears when two timelines share a common node but with different successors.

The shape of the evolutionary maps strongly depends on two main things that we will deal with these issues in the next Sections. The first one is the method used to align the topics and we will discussed the choice we made in Section 4.4. The second one is the algorithm that will be used to determine the timelines and to build the evolutionary maps, we present our new method in Section 4.5.

4.4 Temporal matching

The temporal matching problem correspond to the first challenge that needs to be discuss to construct evolutionary maps. It consists in determining the way the clusters of terms, i.e. the scientific topics, from two following periods will be associated to build a timeline.

4.4.1 Embedding alignment

Problem statement

Our constrain of hierarchies alignment from different periods of time requires to compare both the structure of the quasi-dendrograms and the content of their nodes. To compare the nodes, i.e the topics seen as bags of word vectors, we need a feature that will describe them precisely and the most natural one is to take the barycenter of each

cluster of term vectors. However in order to compare vectors from different time-periods we must ensure that the vectors are aligned to the same coordinate axes. Indeed when training embeddings such as SVD (Levy, Goldberg, and Dagan 2015) or SGNS (Tomáš Mikolov, Sutskever, et al. 2013) on different periods of time nothing guarantees that a same word will have a similar representation as a given dimension of the embedding space has no practical meaning. If the pairwise distances are preserved, it may result in arbitrary transformations of the vector space.

Two main axes of research are at the lead in word vector alignment. They aim to provide the best multi-lingual mapping and to study linguistic changes through time.

Multi-lingual mapping

Several tools are available to map multi-lingual vector such as **VecMap** [1] (Artetxe, Labaka, and Agirre 2018) or **MUSE** [2] (Conneau et al. 2017). Both are the result of improvements over the years since Tomas Mikolov, Le, and Sutskever 2013 first discovered that word embedding spaces present similar structures across languages. It means that if the structure of the embedding spaces are similar, then the distance between terms is similar. And the same way we can query the embedding space to retrieve the word that corresponds to the same relation as between *man* and *king* for the word *woman*, that is *queen*, if the embedding spaces are aligned we could do the same across languages and determine that *reine* is the same to *femme* as *king* is to *man*. In fact aligning the embedding spaces consists in finding the right rotation and translation. Indeed if we distort too much the embedding space, the distances between terms will change as long as the relation between them (see Definition 17). Primary works used a parallel vocabulary as anchor points for the mapping (Faruqui and Dyer 2014, Xing et al. 2015, Ammar et al. 2016, Artetxe, Labaka, and Agirre 2016). Other attempts tried to reduce the size of this "seed" dictionary (Smith et al. 2017, Artetxe, Labaka, and Agirre 2017) but are limited to similar languages. Newer methods explored unsupervised approaches (Cao et al. 2016, M. Zhang et al. 2017) and eventually reached the performances of the supervised methods (Conneau et al. 2017, Artetxe, Labaka, and Agirre 2018).

1. `https://github.com/artetxem/vecmap`
2. `https://github.com/facebookresearch/MUSE`

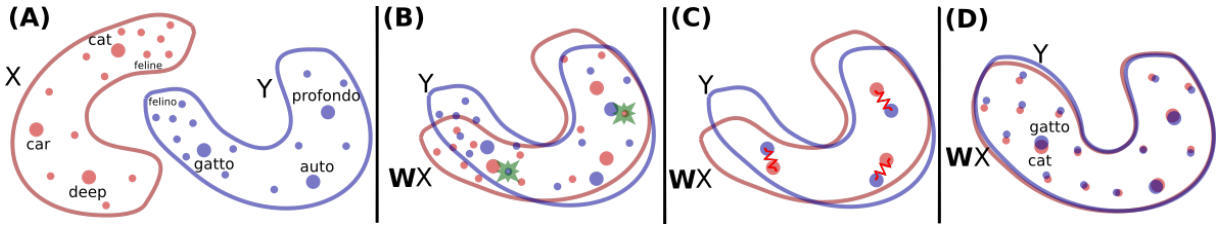


Figure 4.2: Illustration of the MUSE alignment method (Conneau et al. 2017).

Linguistic change study

People also studied linguistic changes by following the evolution of the embedding structures for different periods of time and faced the same problem of embedding alignment. While some used no low-dimensional embedding or performed a local alignment with a linear regression (Kulkarni et al. 2015) other used **orthogonal Procrustes** which determine the best rotational alignment using SVD (Hamilton, Leskovec, and Jurafsky 2016). This Procrustes technique is also part of the supervised learning used for MUSE (Conneau et al. 2017). An even more recent method proposed to simultaneously learn the embeddings and align them across time using time-aware word embeddings (Yao et al. 2018).

Definition 17 (Orthogonal Procrustes problem, Gower, Dijksterhuis, et al. 2004)**.** *The name Procrustes*[3] *refers to a bandit from the Greek mythology who attacked people by stretching them or cutting off their legs, so as to force them to fit the size of an iron bed. The Procrustes analysis consists then in analysing the distribution of a set of shapes.*

To compare the shapes of two or more objects, the objects must be first optimally "superimposed". Procrustes Superimposition (PS) is performed by optimally translating, rotating and uniformly scaling the objects. In other words, both the placement in space and the size of the objects are freely adjusted. The aim is to obtain a similar placement and size, by minimizing a measure of shape difference called the Procrustes distance between the objects.

The Orthogonal Procrustes problem is a method used to find the optimal orthogonal linear transformation for the Procrustes Superimposition of an object with respect to another. Formally, given two matrices A and B, it aims to find the an orthogonal matrix R

3. `https://en.wikipedia.org/wiki/Procrustes_analysis`

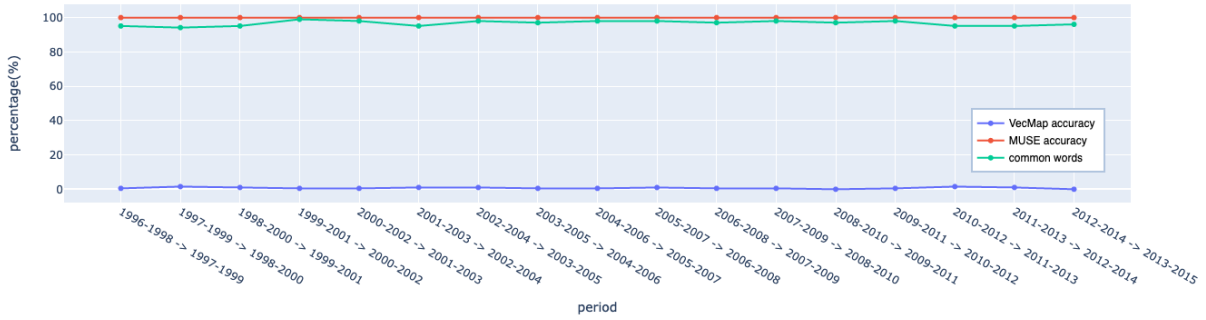


Figure 4.3: Embedding alignment accuracy on the common term dictionary between periods using MUSE and VecMap.

which most closely maps A to B with:

$$R = argmin_{\Omega}||\Omega A - B||_F, \Omega^T\Omega = I \tag{4.1}$$

where $||\cdot||_F$ *is the Frobenius norm that can be applied to matrices.*

Other address the embedding alignment problem as an **Optimal Transport (OT) problem** using the Gromov-Wasserstein distance that measures how distances between pairs of words are mapped across languages (Alvarez-Melis and Jaakkola 2018). This OT method is used in an unsupervised fashion while they also address the orthogonal Procrustes problem as an advantageous formulation to a supervised alignment. Better results occurs on specific tasks while VecMap remains better on average.

Adopted solution

Our task consists in aligning vectors from the same language but from different vector spaces. This is a huge advantage, we saw that if we know corresponding terms it can help the aligning algorithms so we can give all common terms from the consecutive periods as seed dictionary. In fact as we consider overlapping periods, the common terms are a significant part of the vocabularies, more than 90% for the Wiley dataset for all the different vocabulary size from 100 to 1000 terms. With these huge overlaps between the vocabularies we can even use the supervised version of the learning algorithms like *MUSE* and *VecMap* to produce better results.

To validate the alignment of the embedding spaces we can reuse the same common words and determine the proportion of terms that are correctly aligned in the common embedding space. In machine learning, using the same data to train the model and to

test it is a not a good practice as it usually lead to overfitting. However in our case the finality is not to use the aligning models with other unknown word vectors that could be misplaced to any of the embedding spaces so overfitting is not really relevant and we can use the complete set of common words for both training and testing. Figure 4.3 shows the results of different embedding space alignments for consecutive periods of the Wiley dataset based on a vocabulary of 200 terms. As said before we can see that we have more than 90% of the terms in common between the periods and if *MUSE* manages to correctly aligned all of these common terms, *VecMap* poorly performs with only few percent of correct alignments. It might be caused by a bad parametrization but for the rest of the experiments we will use *MUSE* as embedding alignment algorithm.

4.4.2 Hierarchy alignment principle

Now that we determined a method to align the embeddings from different periods of time, we can compute comparable features for each node of the hierarchies and deal with our main problem of hierarchy alignment.

Many methods exists in network alignment as it has many applications in domains like social network study or protein alignment in bioinformatics. Spectral methods such as *FINAL* (S. Zhang and Tong 2016) or *REGAL* (Heimann et al. 2018) fail to deal with very large networks due to matrix diagonalisation which generally has cubic complexity in the number of nodes.

In very recent work, Trung et al. 2020 proposed an embedding-based network alignment model. The idea is to embed nodes of two networks into multi-order feature vectors and then align the networks based on the similarity of their node embeddings. This is realized through the design of a graph convolutional network (GCN) to learn the embeddings. Hence this **GAlign** method works as an unsupervised network alignment without any prior knowledge on the relation of the networks which means no anchor points required.

The output of this algorithm family is a $n \times m$ similarity matrix that represent how good the match is between each node of the source graph (n nodes) and each node of the target graph (m nodes).

There is no method to perfectly address these issues. Whatever methods we chose, there is an inherent uncertainty about the results. Our goal is to build maps that are robust even in the presence of uncertainties about the clusters or their alignments. Also the exploration of choosing the best hierarchy alignment method for our specific task is

not part of this work and any method that produce the same output from the same inputs could replace it. This is no a sealed brick in our framework.

4.5 Evolutionary map construction

In this section we will see the workflow for evolutionary map generation. Let us assume that we would like to analyse a document collection, where the documents are organized to n time periods. Let us also assume that we have constructed a set of quasi-dendrograms $Q = \{Q_1, \ldots, Q_n\}$ for each period for a given merging criterion and the alignment matrices $A = \{A_1, \ldots, A_{n-1}\}$ using the techniques seen in the previous Section. The alignment matrix A_i describes the connections between the concepts of the quasi-dendrograms Q_i and Q_{i+1}.

The construction of evolutionary maps is based on user queries. The query might be a simple query, that is a list of keywords or a list of simple queries, that should lead to the construction of a more complex evolutionary map that is the union of the maps corresponding to the simple queries it is composed of.

4.5.1 Simple evolutionary map

In the following we describe the construction of an evolutionary map related to a simple query. In short we have to identify a topic corresponding to the query in the hierarchies and then to build the associated timelines.

Topic selection

Given a set of words as input query we first need to determine the domain it represents. If one can imagine several possibilities to identify this domain, it will obviously depends on the considered hierarchy Q_s of the set of quasi-dendrograms and the vocabulary V_s it is based on. The idea is to determine the node of the hierarchy that corresponds the most to the set of terms. First we can eliminate the terms of the query that are not in the vocabulary of the hierarchy Q_s. Then we chose to select the smallest node that contains at least the remaining terms of the query. It is given by the formula in Equation 4.2. This way we determine the correct granularity of the identified domain, selecting a similar node

in term of number of elements.

$$c_s = argmin\{|c| \mid c \in Q_s \ \& \ q \cap V_s \subset c\} \tag{4.2}$$

A better interaction could be done by directly allowing the user to use terms only from the union of the vocabularies over all the periods to formulate its query and not terms of his complete choice. Or we can even imagine a tool to navigate in the quasi-dendrogram starting from its root that would allow the user to directly select a node of the hierarchy.

Timeline identification

Then starting from a topic c_i in a quasi-dendrogram Q_i, we can use the alignments $\{A_1, \ldots, A_{n-1}\}$ to build the sequence of evolution of c_i. This offers us several possibilities.

The simplest one consists in following the alignments in both direction, in the future and in the past, by choosing the most similar node of the next quasi-dendrograms given by the alignment matrices. This way the value of similarity between the nodes of the timeline is not important. However we can stop the evolution of the topic in one the directions when the set of terms is completely different from the previous period. Indeed, choosing the maximal value of the alignment matrix will always give us node candidate even when in reality the timeline should be stopped as this topic ceases to appear in the quasi-dendrograms.

$$\begin{aligned} c_{i+1} &= \begin{cases} argmax\{A_i(c_i, .)\} & \text{if } argmax\{A_i(c_i, .)\} \cap c_i \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases} \\ c_{i-1} &= \begin{cases} argmax\{A_{i-1}(., c_i)\} & \text{if } argmax\{A_{i-1}(., c_i)\} \cap c_i \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases} \end{aligned} \tag{4.3}$$

As i represents a period of time we can then consider the timeline t produced by a query q and starting from the period s as the sequence

$$T_{q,s} = ((a, c_a), \ldots, (s-1, c_{s-1}), (s, c_s), (s+1, c_{s+1}), \ldots, (b, c_b))_{1 \leq a \leq b \leq n} \tag{4.4}$$

where a and b represent the respectively the beginning and the end of the timeline either 1 and n if the timeline stops at the extremity of the data or other values in between if an empty set was reached before by following the alignments. An example is given in Figure 4.4. The selected topic, at the fourth period in our example, is highlighted by being circled with a specific color.

Figure 4.4: A timeline of the evolution of the topic *{cell, expression, specific}*.

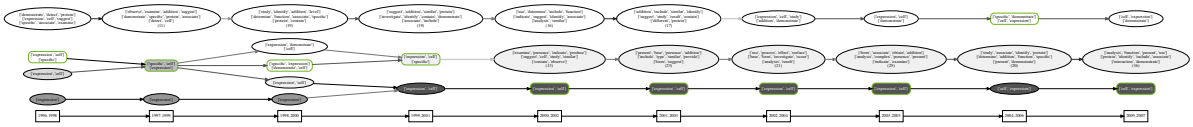


Figure 4.5: An evolutionary map of the evolution of the topic *{cell, expression, specific}*.

A second method to produce a timeline from an topic c_i is to select not the most similar topic according to the alignments but to select all topics with a similarity above a given threshold. We did not explore this approach.

Evolutionary map generation

Changing the period and the quasi-dendrogram in which the starting topic is selected can lead to significant different results in the produced timeline as the selected topic can be completely different. To build a robust map of evolution corresponding the most to a query, an evolutionary map is finally built by union of the timelines generated from all the possible starting periods for the query q.

$$EM_{Q,A}(q) = \{(i, T_{q,i})\}_{1 \leq i \leq n} \tag{4.5}$$

In aggregating the timelines, temporal nodes shared among several lines will be merged in the graph which will produce splits and merges in the structure. Also keeping all the timelines allows to consider nodes and edges with their multiplicity. This multiplicity can be highlighted on the map presented to the expert which will represent robust evolutions of topics. Indeed if starting from different periods of time, the identified topics are the same and evolve in the same way, then we can rely on it compared to a timeline generated from a "noisy" quasi-dendrogram in which the topic identification gave a really different result or the evolution of topic was different.

A recap of the construction is given in Algorithm 2.

Following Figure 4.4 that presented one timeline of evolution the words *{cell, expression, specific}*, Figure 4.5 shows a complete evolutionary map generated from the same query. A darker color on the temporal nodes is used when they are shared among many

Algorithm 2 Simple evolutionary map construction.

Input:
— Quasi-dendrograms $Q = \{Q_1, \ldots, Q_n\}$.
— Alignment matrices $A = \{A_1, \ldots, A_{n-1}\}$.
— Query $q = \{t_1, \ldots, t_m\}$.
Output: Evolutionary map $EM_{Q,A}(q)$.
$i = 1$.
while $i \leq n$ **do**
    Determine the closest match c_i of q in Q_i.
    Follow the alignments A to build a timeline $T_{q,i}$ starting from c_i.
    Add $T_{q,i}$ to the evolutionary map $EM_{Q,A}(q)$.
    $i = i + 1$.
return $EM_{Q,A}(q)$.

timelines.

4.5.2 Complex evolutionary map

As said previously, an evolutionary map can also be build from a more complex query, i.e. from several set of terms, each one representing a simple query.

Algorithm 3 Complex evolutionary map construction.

Input:
— Quasi-dendrograms $Q = \{Q_1, \ldots, Q_n\}$.
— Alignment matrices $A = \{A_1, \ldots, A_{n-1}\}$.
— Query $\{q_1, \ldots, q_k\}$.
Output: Evolutionary map $EM_{Q,A}(\{q_1, \ldots, q_k\})$.
$i = 1$.
while $i \leq k$ **do**
    Build the evolutionary map $EM_{Q,A}(q_i)$.
    Merge $EM_{Q,A}(q_i)$ with the evolutionary map $EM_{Q,A}(\{q_1, \ldots, q_{i-1}\})$.
    $i = i + 1$.
return $EM_{Q,A}(\{q_1, \ldots, q_k\})$.

Presented in Algorithm 3 it works by merging simple evolutionary maps generated from each part of the complex query. In Chapter 2 we defined a method to cut a quasi-dendrogram to a precise level to extract the best cover of k clusters. Then to minimize the impact of choice of terms in a complex query, an expert can directly use the k clusters extracted from a quasi-dendrogram to build a k-component query.

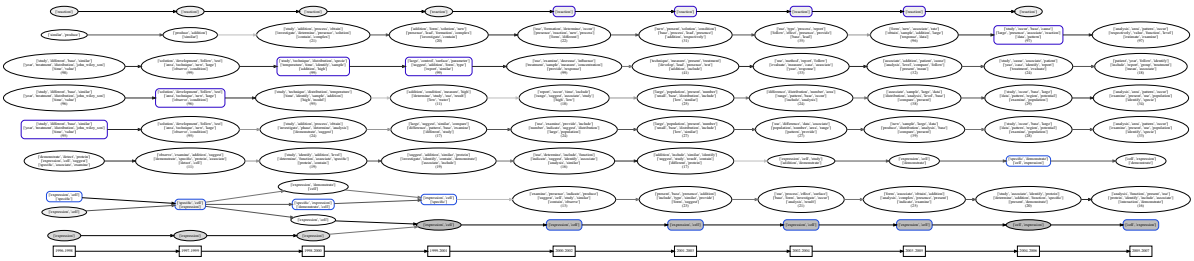


Figure 4.6: An evolutionary map of the evolution of the topics *{cell, expression, specific}* and *{reaction, specific}*.

An example is given in Figure 4.6. It is really difficult to read it correctly in this format as it not possible to zoom in the map to see what inside the nodes. It was already the case for the timeline (Figure 4.4) and the simple evolutionary map (Figure 4.5). What it is important here is to be able to understand the general structure of this objects, identifying merges and splits between the timelines. Bigger Figures are given in 4.7 and to dig into it we will provide notebooks on a git repository as well as the source code so people can play with it an reuse it in their projects if they want.

4.6 Conclusion

In this Chapter we proposed to generate evolutionary maps of scientific domains from a user query in the form of sets of terms representing topics that the user wants to follow. In our framework an evolutionary map is defined as a set of timelines, i.e a set of sequences of topics from consecutive periods. Based on hierarchical representations of keyterms extracted from a corpus of scientific papers sliced in time periods, it works using alignment matrices between following hierarchies.

We detailed a complete workflow that allow this reconstruction and applied our proposal to quasi-dendrograms produced at in the previous Chapters and aligned with the GAlign algorithm. In theory any method used in the workflow could be replaced by an equivalent one, that is one that takes the same input and produce the same type of output. For instance we could use classical dendrograms or a different alignment algorithm. Moreover we defined a probability of evolution that, if used as a threshold, produces more robust evolutionary maps. Only the most taken timelines will be displayed.

As future work, an extensive study can be conducted to properly identify the impacts of the choices made to build the evolutionary maps and find the best possible tuning. Of

course this should be conducted with the help of historians and philosophers of science.

The quasi-dendrogram structure seen in Chapter 2, associated with the similarity seen in Chapter 3 and combined with alignment techniques to build evolutionary maps as presented in Chapter 4 finally achieve our initial goal within the ANR EPIQUE project. In the following Chapter we sum up our work and conclude by presenting general future directions.

4.7 Detailed figures related to the evolutionary maps

Timelines and simple evolutionary maps

(a) A timeline of the evolution of the topic *{cell, expression, specific}*.

(b) An evolutionary map of the evolution of the topic *{cell, expression, specific}*.

Figure 4.7: Left part of the figures.

(a) A timeline of the evolution of the topic *{cell, expression, specific}*.

(b) An evolutionary map of the evolution of the topic *{cell, expression, specific}*.

Figure 4.8: Right part of the figures.

Complex evolutionary maps

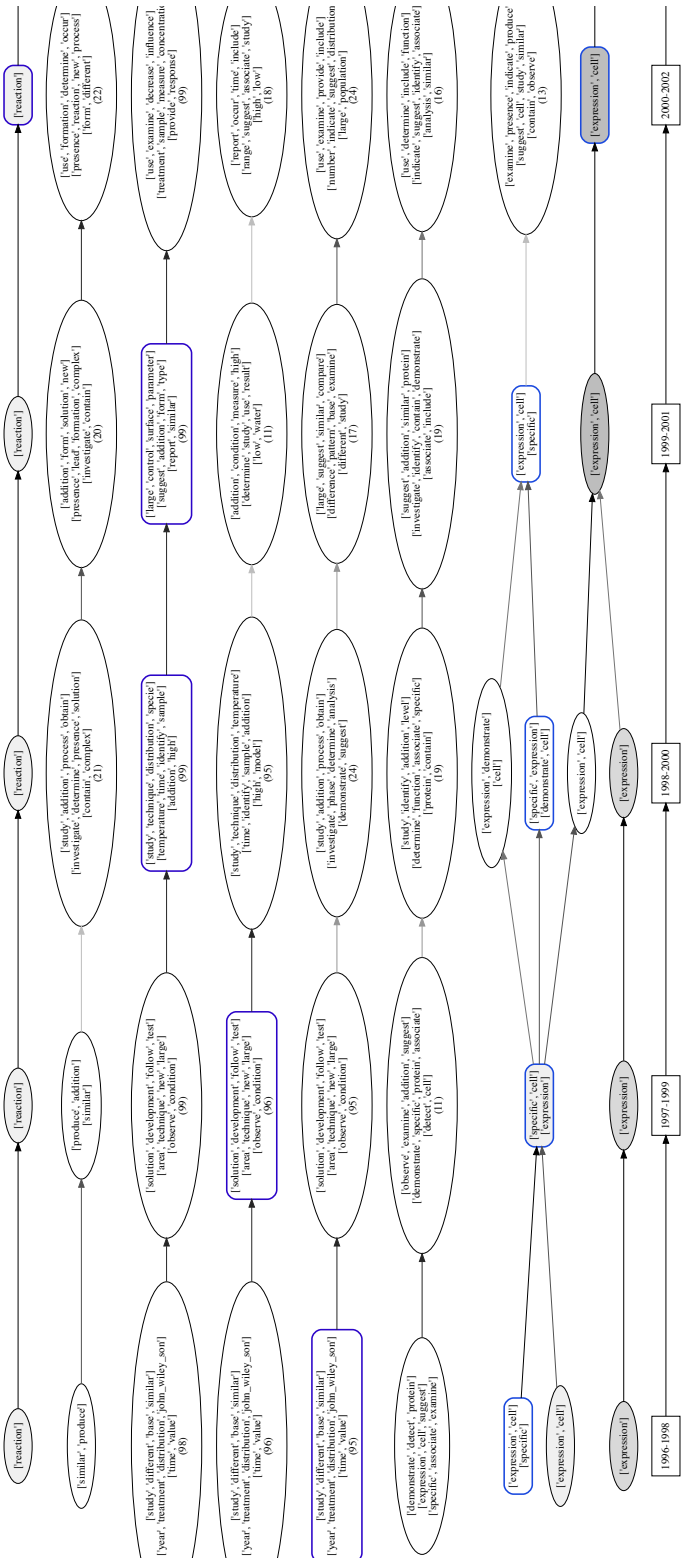


Figure 4.9: An evolutionary map of the evolution of the topics *{cell, expression, specific}* and *{reaction, specific}*. Left part.

Figure 4.10: An evolutionary map of the topics *{cell, expression, specific}* and *{reaction, specific}*. Right part.

Conclusion

Result overview

This thesis is at the crossroads of two major scientific domains, computing and philosophy of science. We developed methods in computer science that have direct use in the work of social scientists. More precisely the presented works aim to provide automatic analysis of scientific publications for quantitative epistemology. Like many other techniques, the end goal is to produce maps of the evolution of scientific fields to help epistemologists determine the mechanisms at play.

We first proposed to enrich the insights on the structure of science with the definition of a new hierarchical structure called a quasi-dendrogram. This structure, a generalization of a simple tree, can be seen as a specific directed acyclic graph. We proposed a framework including a new overlapping hierarchical clustering (OHC) algorithm to generate such hierarchy from a set of scientific papers. This way, at each level, we represent a set of possibly overlapping clusters. If the clusters present in the data show no overlaps, the obtained clusters are identical to the clusters we can compute using classical agglomerative clustering methods. In case of overlapping and nested clusters, however, our method results in a richer representation that can contain relevant information about the structure of the clusters. In fact this algorithm can be applied to any other type of data on which a similarity measure is defined, exactly as for any agglomerative clustering method. By construction we proved theoretical properties of the quasi-dendrograms and showed how the OHC algorithm behaves on synthetic and real data and discussed the scaling issue inherent to this type of algorithm.

One of the major issues was the absence of datasets with a ground truth on the structure of science therein. As many authors, we found ourselves in the incapacity to compare the hierarchies we produced to such benchmarks. Instead we opted for the definition of a similarity measure to be able to compare the quasi-dendrograms between them but also with other classical hierarchical structures like dendrograms. We wanted to be able to highlight the specificities and regularities that could occur in a set of hierarchies produced with different algorithms or with different parameters on the same dataset. To our

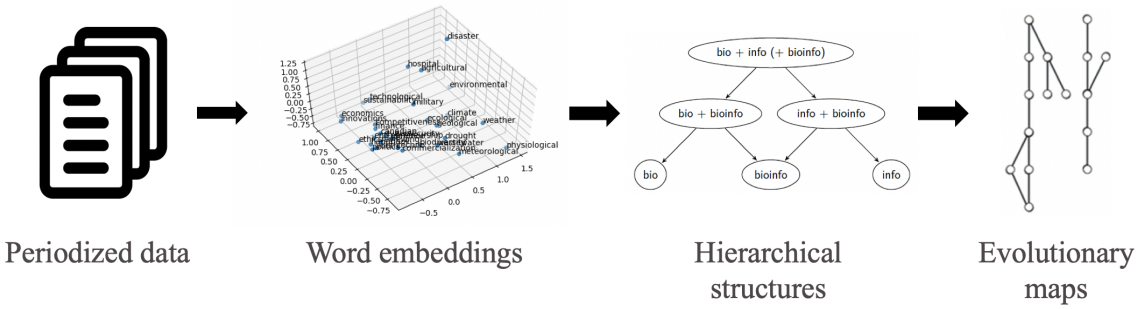


Figure 4.11: New workflow of evolutionary maps reconstruction.

knowledge, very little exists in the literature when the structure becomes more complex than a simple tree. Hence we proposed a new similarity measure that approaches the problem with a level point of view. Indeed, if for two hierarchical structures when we extract a k-cover we obtain similar clusters for any value of k it means that the structures are very similar and conversely. We used this similarity measure to confirm by the experiment that the OHC algorithm acts like a classical single-linkage algorithm when the merging criterion is close to 1 and differ more and more when it decreases. We also proposed a detailed view of the similarity to detect common and different parts of the hierarchies and a weighted variant with the goal to reduce the impact caused by small noise clusters that would persist in the hierarchies. In fact detecting similar parts across several hierarchies, i.e. that are produced for several merging criterion, can be seen as a robustness criterion. This could be explored and combined to the notion of persistence to determine the most robust clusters of the quasi-dendrograms.

Finally we proposed an alternative method to generate evolutionary maps of scientific domains from a user query in the form of sets of terms representing topics that the user wants to follow. In our framework an evolutionary map is defined as a set of timelines, i.e a set of sequences of topics from consecutive periods. Based on hierarchical representations of keyterms extracted from a corpus of scientific papers sliced in time periods, it works using alignment matrices between successive hierarchies. We detailed a complete workflow that allows this reconstruction and applied our proposal to quasi-dendrograms aligned with a specific algorithm. Also we defined a probability of evolution that, if used as a threshold, produces more robust evolutionary maps. Only the most taken timelines will be displayed.

The global workflow (Figure 4.11) that produces evolutionary maps from a set of scientific publications aims to be as standard as possible. It means that the steps are

clearly defined with precise inputs and outputs and in theory any method used in this workflow could be replaced by an equivalent one, that is one that takes the same input and produce the same type of output. For instance we could use classical dendrograms or a different alignment algorithm.

We also created interactive Jupyter notebooks that allows to generate custom quasi-dendrograms and custom evolutionary maps. In fact it aims to provide interactivity during the generation process. These notebooks will be dropped on a git repository as well as the code that will be open sourced.

Future work

The first limitation of this workflow of evolutionary map construction is its evaluation. Indeed it is really hard to determine general properties or characteristics that every evolutionary map should respect. Same as for other methods, the evaluation part is left to the expert and the quality of the maps is based on construction properties specific to the method. The first work that should be done is a proper comparison with other types of evolutionary map. This means to determine a standard workflow in which all the methods could fit and to propose relevant evaluations based on this workflow. It is tricky as all the maps do not serve the same purpose. A first comparison work should be done soon with the partners of the ANR EPIQUE project.

Also to improve the proposed workflow that uses many methods at different steps, all equally important, an extensive study can be conducted to properly identify the impacts of the choices made to build the evolutionary maps and find the best possible tuning. And of course this should be conducted with the help of historians and philosophers of science. To this end, an interaction with the user is required also to analyse the produced structures and interactive navigational tools appear essential in a short term future. One particular goal is to be able to zoom in and out on the structure of science by moving up and down in the hierarchies.

More generally this family of methods based on the content of scientific publication is still limited by the availability of the datasets even with the impulse for open science. Indeed these datasets are limited in terms of diversity of domains they can cover, many disciplines of social science are under-represented and many others are still under paywalls or have their access restricted for some reasons. For instance when a domain like pharmacology is largely financed by private companies, many of its works are patented.

Some large datasets are also restricted in their use and one can often access only to the metadata and not the real content. This is the case for instance for Web Of Science or Wiley where it is not possible to access to the full content of the scientific publications.

Philosophically, to understand the mechanisms of evolution that drive progress and science we should also consider the way science and knowledge are produced. We mentioned in the Introduction the scientific method that implies a peer review and that is an interesting validating process (which is not flawless but the best we have for now) but science is also driven by the way the scientists are financed. Indeed while we cannot blame the private sector, which focuses the research it finances on its own themes, often with a lucrative vision, academic research funding policies, especially in France, increasingly consider knowledge as a commodity like any other in a quest for ever more profitability. The competition at almost all levels of researchers has a strong influence on scientific production, orienting it more and more on fashionable themes (often driven by incentives from outside the scientific world) by strengthening the weight of calls for projects. The logic of evaluation of these same researchers and their laboratories based mainly on the number of publications leads to the infernal spiral of "publish or perish", forcing them to choose the ease of certain themes and abandon other equally promising ones. This generates an over-representation of certain topics in scientific journals and logically skews a study of the evolution of science based on it. Measuring objectively the impact of the key words chosen by funding agencies on the evolution of science could be an interesting project.

BIBLIOGRAPHY

Agrawal, Rakesh et al. (2005), « Automatic subspace clustering of high dimensional data », *in*: *Data Mining and Knowledge Discovery* 11.*1*, pp. 5–33.

Alvarez-Melis, David and Tommi S Jaakkola (2018), « Gromov-wasserstein alignment of word embedding spaces », *in*: *arXiv preprint arXiv:1809.00013*.

Ammar, Waleed et al. (2016), « Massively multilingual word embeddings », *in*: *arXiv preprint arXiv:1602.01925*.

Ankerst, Mihael et al. (1999), « OPTICS: ordering points to identify the clustering structure », *in*: *ACM Sigmod record*, vol. 28, 2, ACM, pp. 49–60.

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre (2016), « Learning principled bilingual mappings of word embeddings while preserving monolingual invariance », *in*: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2289–2294.

— (2017), « Learning bilingual word embeddings with (almost) no bilingual data », *in*: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 451–462.

— (2018), « A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings », *in*: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 789–798.

Bandelt, H-J and Andreas WM Dress (1989), « Weak hierarchies associated with similarity measures—an additive clustering technique », *in*: *Bulletin of mathematical biology* 51.*1*, pp. 133–166.

Beliga, Slobodan, Ana Meštrović, and Sanda Martinčić-Ipšić (2015), « An overview of graph-based keyword extraction methods and approaches », *in*: *Journal of information and organizational sciences* 39.*1*, pp. 1–20.

Bellman, Richard (1966), « Dynamic programming », *in*: *Science* 153.*3731*, pp. 34–37.

Bengio, Yoshua, Réjean Ducharme, et al. (2003), « A neural probabilistic language model », *in*: *Journal of machine learning research* 3.*Feb*, pp. 1137–1155.

Bengio, Yoshua, Yann LeCun, et al. (2007), « Scaling learning algorithms towards AI », *in*: *Large-scale kernel machines* 34.*5*, pp. 1–41.

Bergmanis, Toms and Sharon Goldwater (2018), « Context sensitive neural lemmatization with lematus », *in*: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1391–1400.

Bezdek, James C (2013), *Pattern recognition with fuzzy objective function algorithms*, Springer Science & Business Media.

Blondel, Vincent D et al. (2008), « Fast unfolding of communities in large networks », *in*: *Journal of statistical mechanics: theory and experiment* 2008.*10*, P10008.

Bojanowski, Piotr et al. (2017), « Enriching word vectors with subword information », *in*: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146.

Campello, Ricardo JGB et al. (2015), « Hierarchical density estimates for data clustering, visualization, and outlier detection », *in*: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.*1*, p. 5.

Cao, Hailong et al. (2016), « A distribution-based model to learn bilingual word embeddings », *in*: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 1818–1827.

Chavalarias, David and Jean-Philippe Cointet (2013), « Phylomemetic patterns in science evolution—the rise and fall of scientific fields », *in*: *PloS one* 8.*2*, e54847.

Chen, Chaomei (2004), « Searching for intellectual turning points: Progressive knowledge domain visualization », *in*: *Proceedings of the National Academy of Sciences* 101.*suppl 1*, pp. 5303–5310.

Chinchor, N (1992), *MUC-4 evaluation metrics in Proc. of the Fourth Message Understanding Conference 22–29*.

Conneau, Alexis et al. (2017), « Word Translation Without Parallel Data », *in*: *arXiv preprint arXiv:1710.04087*.

Coscia, Michele et al. (2014), « Uncovering hierarchical and overlapping communities with a local-first approach », *in*: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 9.*1*, pp. 1–27.

Cui, Weiwei et al. (2011), « Textflow: Towards better understanding of evolving topics in text », *in*: *IEEE transactions on visualization and computer graphics* 17.*12*, pp. 2412–2421.

Davies, David L and Donald W Bouldin (1979), « A cluster separation measure », *in*: *IEEE transactions on pattern analysis and machine intelligence 2*, pp. 224–227.

Defays, Daniel (1977), « An efficient algorithm for a complete link method », *in*: *The Computer Journal* 20.*4*, pp. 364–366.

Dias, Laércio et al. (2018), « Using text analysis to quantify the similarity and evolution of scientific disciplines », *in*: *Royal Society open science* 5.*1*, p. 171545.

Diatta, Jean (1997), « Dissimilarités multivoies et généralisations d'hypergraphes sans triangles », *in*: *Mathématiques et sciences humaines* 138, pp. 57–73.

Diday, Edwin (1984), « Une représentation visuelle des classes empiétantes: les pyramides », *in*.

Diestel, Reinhard (2005), « Graph theory. 2005 », *in*: *Grad. Texts in Math* 101.

Dumais, Susan T (2004), « Latent semantic analysis », *in*: *Annual review of information science and technology* 38.*1*, pp. 188–230.

Dunn, Joseph C (1974), « Well-separated clusters and optimal fuzzy partitions », *in*: *Journal of cybernetics* 4.*1*, pp. 95–104.

Ester, Martin et al. (1996), « A density-based algorithm for discovering clusters in large spatial databases with noise. », *in*: *Kdd*, vol. 96, 34, pp. 226–231.

Faruqui, Manaal and Chris Dyer (2014), « Improving vector space word representations using multilingual correlation », *in*: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 462–471.

Fellbaum, Christiane (2017), « WordNet: An electronic lexical resource », *in*: *The Oxford Handbook of Cognitive Science*, Routledge, pp. 301–314.

Firoozeh, Nazanin et al. (2020), « Keyword extraction: Issues and methods », *in*: *Natural Language Engineering* 26.*3*, pp. 259–291.

Fowlkes, Edward B and Colin L Mallows (1983), « A method for comparing two hierarchical clusterings », *in*: *Journal of the American statistical association* 78.*383*, pp. 553–569.

Frank, Eibe et al. (1999), « Domain-specific keyphrase extraction », *in*: *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pp. 668–673.

Garfield, Eugene (1964), « " Science Citation Index"-A New Dimension in Indexing », *in*: *Science* 144.*3619*, pp. 649–654.

Globerson, Amir et al. (2007), « Euclidean embedding of co-occurrence data », *in*: *Journal of Machine Learning Research* 8.*Oct*, pp. 2265–2295.

Goldberg, Yoav and Omer Levy (2014), « word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method », *in*: *arXiv preprint arXiv:1402.3722*.

Golub, Gene H and Christian Reinsch (1971), « Singular value decomposition and least squares solutions », *in*: *Linear Algebra*, Springer, pp. 134–151.

Gower, John C, Garmt B Dijksterhuis, et al. (2004), *Procrustes problems*, vol. 30, Oxford University Press on Demand.

Grineva, Maria, Maxim Grinev, and Dmitry Lizorkin (2009), « Extracting key terms from noisy and multitheme documents », *in*: *Proceedings of the 18th international conference on World wide web*, pp. 661–670.

Gutmann, Michael U and Aapo Hyvärinen (2012), « Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics », *in*: *Journal of Machine Learning Research* 13.*Feb*, pp. 307–361.

Hakkani-Tür, Dilek Z, Kemal Oflazer, and Gökhan Tür (2002), « Statistical morphological disambiguation for agglutinative languages », *in*: *Computers and the Humanities* 36.*4*, pp. 381–410.

Hamilton, William L, Jure Leskovec, and Dan Jurafsky (2016), « Diachronic word embeddings reveal statistical laws of semantic change », *in*: *arXiv preprint arXiv:1605.09096*.

Harris, Zellig S (1954), « Distributional structure », *in*: *Word* 10.*2-3*, pp. 146–162.

Heimann, Mark et al. (2018), « Regal: Representation learning-based graph alignment », *in*: *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 117–126.

Hoffmann, Oliver and Florian Reitz (Apr. 2018), *hdblp: historical data of the dblp collection*, dblp is a joint project of the University of Trier, Germany and the Schloss Dagstuhl – Leibniz Center for Informatics, Wadern, Germany. Former and current members of the team are listed on http://dblp.org/db/about/team.html ., Zenodo, DOI: 10.5281/zenodo.1213051, URL: https://doi.org/10.5281/zenodo.1213051.

Hu, Xinghua and Bin Wu (2006), « Automatic keyword extraction using linguistic features », *in*: *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*, IEEE, pp. 19–23.

Huang, Chong et al. (2006), « Keyphrase extraction using semantic networks structure analysis », *in*: *Sixth International Conference on Data Mining (ICDM'06)*, IEEE, pp. 275–284.

Hull, David L (1988), *Science as a Process: An Evolutionary Account of the Social and Conceptual Development of Science*.

Jaccard, Paul (1901), « Étude comparative de la distribution florale dans une portion des Alpes et des Jura », *in*: *Bull Soc Vaudoise Sci Nat* 37, pp. 547–579.

Jeantet, Ian, Zoltán Miklós, and David Gross-Amblard (2020), « Overlapping Hierarchical Clustering (OHC) », *in*: *International Symposium on Intelligent Data Analysis*, Springer, pp. 261–273.

Jones, Karen Sparck (1972), « A statistical interpretation of term specificity and its application in retrieval », *in*: *Journal of documentation*.

Krapivin, Mikalai et al. (2010), « Keyphrases extraction from scientific documents: improving machine learning approaches with natural language processing », *in*: *International Conference on Asian Digital Libraries*, Springer, pp. 102–111.

Kuhn, Thomas S (1962), *The Structure of Scientific Revolutions*.

Kulkarni, Vivek et al. (2015), « Statistically significant detection of linguistic change », *in*: *Proceedings of the 24th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, pp. 625–635.

Lakatos, Imre (1976), « Falsification and the methodology of scientific research programmes », *in*: *Can theories be refuted?*, Springer, pp. 205–259.

Lavelli, Alberto, Fabrizio Sebastiani, and Roberto Zanoli (2004), « Distributional term representations: an experimental comparison », *in*: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, ACM, pp. 615–624.

Lebret, Rémi and Ronan Collobert (2013), « Word emdeddings through hellinger PCA », *in*: *arXiv preprint arXiv:1312.5542*.

Levy, Omer, Yoav Goldberg, and Ido Dagan (2015), « Improving distributional similarity with lessons learned from word embeddings », *in*: *Transactions of the Association for Computational Linguistics* 3, pp. 211–225.

Li, Yitan et al. (2015), « Word embedding revisited: A new representation learning and explicit matrix factorization perspective », *in*: *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Liu, Haitao and Fengguo Hu (2008), « What role does syntax play in a language network? », *in*: *EPL (Europhysics Letters)* 83.*1*, p. 18002.

Luhn, Hans Peter (1957), « A statistical approach to mechanized encoding and searching of literary information », *in*: *IBM Journal of research and development* 1.*4*, pp. 309–317.

Lund, Kevin and Curt Burgess (1996), « Producing high-dimensional semantic spaces from lexical co-occurrence », *in*: *Behavior research methods, instruments, & computers* 28.*2*, pp. 203–208.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2009), *Introduction to information retrieval*, Cambridge, UP.

Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz (1993), « Building a large annotated corpus of English: The Penn Treebank », *in*.

McCormick, Chris (2016), *Word2vec tutorial-the skip-gram model*.

McInnes, Leland and John Healy (2017), « Accelerated Hierarchical Density Based Clustering », *in*: *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, IEEE, pp. 33–42.

Mihalcea, Rada and Paul Tarau (2004), « Textrank: Bringing order into text », *in*: *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404–411.

Mikolov, Tomas, Quoc V Le, and Ilya Sutskever (2013), « Exploiting similarities among languages for machine translation », *in*: *arXiv preprint arXiv:1309.4168*.

Mikolov, Tomáš, Kai Chen, et al. (2013), « Efficient estimation of word representations in vector space », *in*: *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomáš, Martin Karafiát, et al. (2010), « Recurrent neural network based language model », *in*: *Eleventh Annual Conference of the International Speech Communication Association*.

Mikolov, Tomáš, Stefan Kombrink, et al. (2011), « Extensions of recurrent neural network language model », *in*: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, IEEE, pp. 5528–5531.

Mikolov, Tomáš, Ilya Sutskever, et al. (2013), « Distributed representations of words and phrases and their compositionality », *in*: *Advances in neural information processing systems*, pp. 3111–3119.

Mnih, Andriy and Geoffrey E Hinton (2009), « A scalable hierarchical distributed language model », *in*: *Advances in neural information processing systems*, pp. 1081–1088.

Mnih, Andriy and Yee Whye Teh (2012), « A fast and simple algorithm for training neural probabilistic language models », *in*: *arXiv preprint arXiv:1206.6426*.

Morin, Frederic and Yoshua Bengio (2005), « Hierarchical Probabilistic Neural Network Language Model. », *in*: *Aistats*, vol. 5, Citeseer, pp. 246–252.

Müller, Thomas et al. (2015), « Joint lemmatization and morphological tagging with lemming », *in*: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2268–2274.

Navigli, Roberto (2009), « Word sense disambiguation: A survey », *in*: *ACM computing surveys (CSUR)* 41.*2*, pp. 1–69.

Newman, Mark EJ and Michelle Girvan (2004), « Finding and evaluating community structure in networks », *in*: *Physical review E* 69.*2*, p. 026113.

Ohsawa, Yukio, Nels E Benson, and Masahiko Yachida (1998), « KeyGraph: Automatic indexing by co-occurrence graph based on building construction metaphor », *in*: *Proceedings IEEE International Forum on Research and Technology Advances in Digital Libraries-ADL'98-*, IEEE, pp. 12–18.

Paice, Chris D (1990), « Another stemmer », *in*: *ACM Sigir Forum*, vol. 24, 3, ACM New York, NY, USA, pp. 56–61.

Palla, Gergely et al. (2005), « Uncovering the overlapping community structure of complex networks in nature and society », *in*: *nature* 435.*7043*, p. 814.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014), « Glove: Global vectors for word representation », *in*: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Popper, Karl (1963), *Conjectures and refutations: The growth of scientific knowledge*, routledge.

Porter, Martin F (2001), *Snowball: A language for stemming algorithms*.

Qu, Jun et al. (2007), « A hierarchical clustering based on overlap similarity measure », *in*: *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, vol. 3, IEEE, pp. 905–910.

Rand, William M (1971), « Objective criteria for the evaluation of clustering methods », *in*: *Journal of the American Statistical association* 66.*336*, pp. 846–850.

Robertson, Stephen and Hugo Zaragoza (2009), *The probabilistic relevance framework: BM25 and beyond*, Now Publishers Inc.

Rosvall, Martin and Carl T Bergstrom (2010), « Mapping change in large networks », *in*: *PloS one* 5.*1*, e8694.

Rousseeuw, Peter J (1987), « Silhouettes: a graphical aid to the interpretation and validation of cluster analysis », *in*: *Journal of computational and applied mathematics* 20, pp. 53–65.

Shahaf, Dafna, Carlos Guestrin, and Eric Horvitz (2012), « Trains of thought: Generating information maps », *in*: *Proceedings of the 21st international conference on World Wide Web*, pp. 899–908.

Shahaf, Dafna, Jaewon Yang, et al. (2013), « Information cartography: creating zoomable, large-scale maps of information », *in*: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 1097–1105.

Shen, Huawei et al. (2009), « Detect overlapping and hierarchical community structure in networks », *in*: *Physica A: Statistical Mechanics and its Applications* 388.*8*, pp. 1706–1712.

Sibson, Robin (1973), « SLINK: an optimally efficient algorithm for the single-link cluster method », *in*: *The computer journal* 16.*1*, pp. 30–34.

Smith, Samuel L et al. (2017), « Offline bilingual word vectors, orthogonal transformations and the inverted softmax », *in*: *arXiv preprint arXiv:1702.03859*.

Sterckx, Lucas et al. (2016), « Supervised keyphrase extraction as positive unlabeled learning », *in*: *EMNLP2016, the conference on empirical methods in natural language processing*, pp. 1–6.

Trung, Huynh Thanh et al. (2020), « Adaptive Network Alignment with Unsupervised and Multi-order Convolutional Networks », *in*: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, IEEE, pp. 85–96.

Van Eck, Nees Jan and Ludo Waltman (2011), « Text mining and visualization using VOSviewer », *in*: *arXiv preprint arXiv:1109.2058*.

Ward Jr, Joe H (1963), « Hierarchical grouping to optimize an objective function », *in*: *Journal of the American statistical association* 58.*301*, pp. 236–244.

Weaver, Warren (1955), « Translation », *in*: *Machine translation of languages* 14.*15-23*, p. 10.

Whissell, John S and Charles LA Clarke (2011), « Improving document clustering using Okapi BM25 feature weighting », *in*: *Information retrieval* 14.*5*, pp. 466–487.

Xing, Chao et al. (2015), « Normalized word embedding and orthogonal transform for bilingual word translation », *in*: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1006–1011.

Yang, Jaewon and Jure Leskovec (2013), « Overlapping community detection at scale: a nonnegative matrix factorization approach », *in*: *Proceedings of the sixth ACM international conference on Web search and data mining*, ACM, pp. 587–596.

Yao, Zijun et al. (2018), « Dynamic word embeddings for evolving semantic discovery », *in*: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, ACM, pp. 673–681.

Zhang, Meng et al. (2017), « Adversarial training for unsupervised bilingual lexicon induction », *in*: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1959–1970.

Zhang, Si and Hanghang Tong (2016), « Final: Fast attributed network alignment », *in*: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1345–1354.

Zhou, Xu et al. (2017), « A density based link clustering algorithm for overlapping community detection in networks », *in*: *Physica A: Statistical Mechanics and its Applications* 486, pp. 65–78.

Zimek, Arthur, Erich Schubert, and Hans-Peter Kriegel (2012), « A survey on unsupervised outlier detection in high-dimensional numerical data », *in*: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 5.5, pp. 363–387.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Analyses hiérarchiques et temporelles de corpora scientifiques vues comme outils pour l'histoire des sciences

Mot clés : Carte d'évolution, Quasi-dendrogramme, Regroupement Hiérarchique Chevauchant, Alignement de hiérarchies

Résumé :

Cette thèse vise à fournir une analyse automatique de publications scientifiques pour l'épistémologie quantitative. L'objectif final est de produire des cartes d'évolution des domaines scientifiques pour aider les épistémologues à déterminer les mécanismes en jeu et ce à partir du texte brut des publications. Nous proposons d'abord d'enrichir les connaissances sur la structure de la science à l'aide d'une nouvelle structure hiérarchique appelée quasi-dendrogramme qui peut être vue comme un graphe acyclique dirigé spécifique. Nous proposons un cadre d'étude comprenant un nouvel algorithme de regroupement hiérarchique chevauchant (OHC) afin de générer une telle hiérarchie à partir du texte d'articles scientifiques. L'un des problèmes majeurs est l'absence de vérité terrain. Nous proposons donc une nouvelle mesure de similarité qui compare des hiérarchies en estimant la correspondance existante entre niveaux de même taille. Enfin, nous proposons une méthode alternative pour générer des cartes évolutives de domaines scientifiques à partir de requêtes. Une carte évolutive est définie comme un ensemble de chronologies déterminées en suivant un alignement de hiérarchies de périodes consécutives. Nous avons également défini une probabilité d'évolution qui utilisée comme un seuil produit des cartes évolutives plus robustes.

Title: Hierarchical and temporal analysis of scientific corpora as tools for the history of science

Keywords: Evolutionary map, Quasi-dendrogram, Overlapping Hierarchical Clustering, Hierarchy alignment

Abstract:

This thesis aims to provide automatic analysis of the raw text of scientific publications for quantitative epistemology. The final goal is to produce maps of evolution of scientific domains to help epistemologists to determine the mechanisms that are at stake. We first propose to enrich the insights on the structure of science with a new hierarchical structure called a quasi-dendrogram that can be seen as a specific directed acyclic graph. We propose a framework including a new overlapping hierarchical clustering (OHC) algorithm to generate such hierarchy from the text of scientific papers. One of the major issues was the absence of ground truth. Hence we propose a new similarity measure that compares hierarchies by estimating the matching of same size levels. Finally we propose an alternative method to generate evolutionary maps of scientific domains from a user query. An evolutionary map is defined as a set of timelines determined in following aligned hierarchies from consecutive periods. We defined a probability of evolution that, if used as a threshold, produces more robust evolutionary maps.