Connecting graphs to machine learning

Gaëlle Candel

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à École Normale Supérieure

Connecting Graphs to Machine Learning

Soutenue par
Gaëlle Candel
Le 10 mars 2022

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité
Informatique

Composition du jury :

Matthieu Latapy Directeur de recherche, CNRS, LIP6	*Président, Rapporteur*
Marc Lelarge Directeur de recherche, INRIA	*Examinateur*
Pr. Amaury Habrard Université Jean Monnet, Saint-Etienne	*Examinateur*
Pr. Stefan Bruckner Université de Bergen, Norvège	*Rapporteur*
Pr. Luca Maria Aiello Université de Copenhague, Danemark	*Examinateur*
Pr. Anastasia Bezerianos LRI-Université Paris-Saclay	*Examinatrice*
Pr. David Naccache École Normale Supérieure	*Directeur de thèse*

ENS | PSL

Remerciements

La thèse a été pour moi un travail singulier, me permettant de faire maturer mes idées. Bien que ce soit un travail individuel, cette thèse n'aurait pas pu aboutir sans le soutient direct et indirect de grand nombre de personnes pour me permettre de surmonter les différents obstacles.

En premier lieu, je pense à Xavier Rival de l'équipe ANTIQUE du département d'informatique de l'ENS, pour sa bienveillance et sa patience, les discussions au quotidien en tout genre, et son support sur les points administratifs. Bien évidemment, je pense aussi aux autres membres permanents de l'équipe, Jérôme et Cezara, ainsi qu'à mes differents (ex)-co-thésards, en particulier à Marc Chevalier.

En second lieu, il y a les personnes de l'INRIA et de l'ENS qui se sont mobilisées pour m'aider à changer d'encadrement de thèse. Ici, je remercie Pierre Gaillard pour m'avoir encadré de façon temporaire, me permettant de me reconnecter aux abstractions.

Après un redémarrage à mi-parcours, la thèse recommence à Ingenico où l'environnement fut propice pour l'aboutissement de cette thèse. J'y remercie mon directeur de thèse David Naccache, pour son encadrement tout au long de cette deuxième partie de thèse, et pour les discussions scientifiques sur des sujets techniques divers et variés.

Il ne faut pas oublier les amis et la famille, qui permettent de se construire en dehors du milieu professionnel, malgré les sacrifices temporels exigés par la thèse. Au hacker-space familial, il y a Ludovic Noury et ses cours de FPGA et Peregrine et son Amiga qui ont été présents tout au long de ces différentes années. Et bien évidemment à mon conjoint Emmanuel Malige, pour son amour, son soutient, et me challenger au quotidien.

Table of Contents

CHAPTER 1

Introduction

1.1 Graphs are everywhere

1.1.1 Real-World Graphs

Graphs are everywhere. They help us to find the shortest path matching our constrains using transportation networks. They allow us to send a message to the correct recipient without testing all the million other addresses in a communication network like the Internet. They benefit to supply chain networks by optimizing the process from end-to-end to reduce stocks without shortage. They permit the identification of a bottleneck in power-network to prevent blackout in the event of inclement weather. They support decentralization with collaborative databases, as in Peer-to-Peer file sharing systems or Blockchains. They help to model disease propagation under different network topologies and update rules in epidemiological networks. They allow studying virality of information over social networks. They organize peers into a swarm acting without central entities by deciding collectively locally such as in autonomous vehicles. They allow controlling a large set of machines with a single computer, simplifying companies' server management while allowing to build and control Botnets to overwhelm some services making them unavailable. They are tools to find "items you may like" without information about your taste or identity by exploiting users with a similar history in recommender systems. They support the discovery of new pathways to synthesize a new molecule and identify by-products using chemical networks. All these problems are represented by a graph, a set of nodes or entities connected to each other by edges or links representing the entities' relationships.

1.1.2 Graph Categories

All these graphs differ by the entities they represent. However the main diversity driver is the edges' type. Edges can be symmetric or directed indicating a form of subordination between the parent and child nodes. A weight can be associated with an edge representing a similarity, probability, edge strength, capacity, distance, or any other real value depending on the use-case. Weighs can evolve over time, encoding time and duration of interactions between entities. Finally, edges can be labeled, precisely characterizing the relationships and adding types to the graph.

Nodes can also be used to refine the graph classification. In the simplest type of graph, nodes have all the same type, but are distinguished by their attributes. In k-mode networks, k different types of entities coexist as in a knowledge graph where places, people, dates, and events occur in the same graph.

The different types of graphs make the richness of this field, leading to a variety of algorithms adapted to the graph particularities.

1.2 Graph and Tabular Data

1.2.1 Dissimilar Structures, Similar Strategies

Graph and tabular data are two types of structured data. Machine Learning (ML) algorithms are specific to one or the other class and cannot be transposed directly to the other category. Out of the difference between data structure, a main differentiating point is how an item is considered. When training an algorithm on tabular data, items can be selected at random to form a subset of smaller size, reducing the training complexity. Items are replaceable and exchangeable. Sampling does not affect algorithms' outcome as long as the subset has a distribution relatively close to the whole set. On a graph, random sampling cannot be done. Some nodes are central and are essential to the graph structure while others are weakly connected and unimportant. The connected structure would be denatured by deleting these central items and the removal of too many edges.

Despite the dissimilarity between graphs and tabular data, the types of problems solved by ML algorithms are similar for both. There exist *supervised* algorithms that try to excel on a particular task with the help of feedback data, and *unsupervised* algorithms where the goal is to identify regularities in the data. There is a philosophical difference where the former focuses on a defined task while the other explores data usability for new applications. In the *unsupervised* category, there is the problem of group identification, which applies to tabular data under the name of clustering, and graphs under the name of community detection. This task is of main interest as it helps discover natural segmentation that is not straight evidence to humans.

1.2.2 Exploiting Graphs under a Matrix Representation

A graph can be converted into an adjacency matrix to apply tabular algorithms on it. For a graph with n items, the adjacency matrix A has a size of $n \times n$, and the entry $A_{i,j}$ is non-zero if i is connected to j, the value corresponding to the weight of the edge or 1 in the case of an unweighted graph. This representation is practical for using tabular algorithms, but can only be used on small graphs as the required size grows quadratically with the number of nodes.

The distribution of tabular data is commonly studied by plotting one variable against another, allowing one to visualize the distribution density on a 2D map. Unfortunately, graphs cannot be projected easily on a 2D space without having crossing edges and distorting edges' lengths. Under the adjacency form, spectral decomposition algorithms such as SVD can be employed to extract the main graph components and identify groups. The quality of the results obtained with this method is impacted by the sparsity of the graph. Despite many nodes, a node is often connected on average to very few nodes, leading to an adjacency matrix mostly made of zeros. With a larger sparsity, the signals are weaker, decreasing the ability of spectral algorithms to identify them.

The main obstacles of tabular data are missing values that prevent the use of an algorithm on them, and outliers, which disturb the data distribution. A pre-processing step is often done to remove items with missing values or fill blanks with data imputation methods. During the pre-processing steps, outliers are often isolated and discarded. Then, data are normalized to give an equal contribution to all features. The missing information also exists in graphs, where some edges might be missing but their absence does not impact the graph processing. The main concerns in real-world graphs are completely different and related to data distribution. Tabular data distribution is often assumed to follow a Gaussian law, while the distribution of nodes' degree follows a power-law. A few nodes are highly connected in this situation while most are weakly connected to the overall structure. While tabular algorithms rarely deal with extreme values, this is the norm in graphs.

In fine, tabular algorithms can hardly be used directly on real-world graphs because of the structural differences. Moreover, graphs do not verify the distribution hypothesis that tabular algorithms assume; therefore, there is no guarantee of the algorithms' outcomes.

1.3 Thesis Organization

This thesis is decomposed into three parts, each dedicated to a particular type of graph. Part I: Identifying Research Communities is specific to citation graphs, which belong to the directed acyclic graph category. Part II: Clustering Sparse Bipartite Graphs addresses the co-clustering of bipartite graphs to identify thematic groups. Last, part III: Computation under Noisy Condition exploits k-nearest neighbors graphs to improve classification accuracy of an ensemble. The problem of each part is detailed in the following paragraphs.

1.3.1 Helping Scientific Watch by Automated Bibliography Analysis

In the life of a researcher, one of the most time-consuming tasks is making the bibliography. A researcher must read many papers to understand the different concepts and challenges of a field. Unfortunately, the amount of literature is too large to be read and understood within human life. Therefore, the researcher has only time to read a few papers selected using a heuristic, exploiting the popularity or citation count, to consider the best known.

Over time, the researcher needs to keep himself up to date about the research topics' evolution and the progress made by the community. Good and great papers are hardly identifiable in their young years as newly published papers have very few citations at the start. The researcher cannot rely on its peers to highlight the most promising papers with citations. Therefore, the researcher must spend some time reading them, making the discovery of emerging trends within its field time-consuming.

The monitoring of recent works is essential for an author. First, before submitting an idea, it must check if its idea is new by looking at other works proposing a similar approach that does not already exist. Next, the bibliography is necessary to set up the scope and support the paper's argumentation. Finally, the inclusion of recent works is essential to demonstrate the current interest of the community for the problem addressed.

A researcher and its research topic focus on a small part of the research landscape. At an institution or company level, there is the problem of scientific and technical watch. Managerial teams need the big picture of what is going on to create new teams that will work on new or yet uncovered topics, and allocate more resources on growing trends.

The evolution of the scientific literature can be studied by looking at the underlying citation graph, where nodes represent papers and edges citation relationships. The monitoring of trends starts with their identification and delineation using community detection algorithms. Then, each trend can be characterized and tracked over time, to identify hot-topics, notorious papers, and people working on the subject for future collaboration.

Chapter 2 (Improving Coupling Metrics with Deep Neighborhood for Local Map of Science) presents a method projecting the citation graph into a 2D map, allowing the identification of research communities using usual clustering algorithms. Next, chapter 3 (Index t-SNE: Tracking Dynamics of High-Dimensional Datasets with Coherent Embeddings) extends the t-SNE algorithm, used to project the citation graph in chapter 2, to the temporal domain, granting the monitoring of research communities' growth over time.

1.3.2 Clustering Sparse Bipartite Graph

A bipartite graph is a special type of 2-mode graph where only nodes of opposite types can connect. This type of graph is very common in everyday life, where people interact in an individualized way with items, such as in a physical (or digital) library with books, an e-commerce platform with goods, a music player with song and podcasts, or a streaming platform

with videos. All the interactions can be represented by pairs `(user, item)`, where the `item` can be any object. The same representation can describe resources (textual document, video, image, music, etc.) with keywords where the association is represented by pairs `(resource, keyword)`. Collaborative filtering recommender systems take advantage of these bipartite graphs to recommend items to a user without knowing anything about the user nor the item. Clustering these bipartite graphs allow identifying typical user profiles, words occurring in a similar context, related products with the same utility and resources with similar content.

We propose in part II clustering algorithms for bipartite graphs addressing sparsity in two different manners. First, in chapter 4 (Tagged Documents Co-Clustering), we propose a hierarchical co-clustering algorithm where the end-user is free to select the final number of clusters. This approach focuses on building groups with homogeneous content and similar size to avoid an imbalanced clustering. Second, chapter 5 (Co-Embedding Bipartite Graphs) also presents a co-clustering algorithm; nonetheless the number of clusters is automatically inferred. This approach exploits graph projections to measure node relatedness, tackling the problem of sparsity.

1.3.3 Collaboration under Noisy Conditions

The last part of this thesis (part III) addresses collaboration over classification and clustering tasks. Traditional ensemble learning methods try to correct labels by using multiple diverse classifiers. The gain of accuracy obtained with these methods depends on the number of peers and their quality. On peer-to-peer network, the presence of malicious peers would disturb the ensemble quality by sending corrupted results. Error-correction algorithms are necessary when the identification of malicious peers is impossible.

Errors are corrected by following general assumptions made by classification algorithms. A classifier assumes that items' classes can be delineated with particular boundaries. The shape and the way to search for the boundaries depend on the algorithm used. These algorithms are capable of very good performances, and are often accurate in very dense areas. However, they are more likely to make errors near a class's boundary. Chapter 6 (Noise-Resilient Ensemble Learning using Evidence Accumulation) proposes an ensemble learning method exploiting the k-nearest neighbors graph to correct labels close to a boundary.

Part I

Identifying Research Communities

CHAPTER 2

Improving Coupling Metrics with Deep Neighborhood for Local Map of Science

Bibliographic and co-citation coupling are two analytical methods widely used to measure the degree of similarity between scientific papers. These approaches are intuitive, easy to put into practice, and computationally cheap. Moreover, they have been used to generate a map of science, allowing visualizing research field interactions. Nonetheless, these methods do not work unless two papers share a standard reference, limiting the two papers' usability with no direct connection. In this work, we propose to extend bibliographic coupling to the deep neighborhood, by using graph diffusion methods. This method allows defining similarity between any two papers, making it possible to generate a local map of science, highlighting field organization.

2.1 Background

2.1.1 History and Usage of Coupling Metrics

Historiographic mapping has been introduced by Garfield [1], retracing the connections between the different discoveries around DNA. This mapping of historical events was made possible thanks to Bibliographic Coupling (BC), introduced by [2], and Co-citation Coupling (CC), introduced by [3]. Both methods use the same approach: two documents are similar if they share at least one cited or citing a paper in common. The similarity score is a function of the overlap between the sets of references or referees. These coupling methods have been used extensively for many purposes: to create a map of science, to study relationships between universities [4], to analyze co-authorship networks [5], or to retrace phylogeny of science [6].

2.1.2 Formal Representation of Citation Graphs

Directed Acyclic Graph: A citation graph can be represented as a directed acyclic graph (DAG) $G = (V, E)$, with $E \subseteq V \times V$, where vertices V represent published papers and edges E the citation relationships, produced from the most recent paper to the oldest one. The formed graph corresponds to a DAG, as it is not possible to find a path from a node to itself, as the edges are directed toward the past.

References and Citations The reference set is denoted $R(a) = \{b|\ (a, b) \in E\}$ for a document a and its citation set is denoted similarly as $C(a) = \{b|(b, a) \in E\}$. This set corresponds to all reachable items at a distance 1 from the currently studied paper a and can be denoted $R^1(a) = R(a)$. The reference set can be recursively extended to indirectly referenced items reachable within k steps from a as $R^k(a) = \bigcup_{b \in R^1(a)} R^{k-1}(b) \cup R^1(a)$. The citation set can be extended the same way as $C^k(a) = \bigcup_{b \in C^1(a)} C^{k-1}(b) \cup C^1(a)$.

Couplings Two papers are coupled if they share some of their neighbors. For BC, the set of papers coupled with a is denoted $BC(a) = \bigcup_{b \in R(a)} C(b)$, while $CC(a) = \bigcup_{b \in C(a)} R(a)$ for co-citation. The coupling can be extended by including indirect neighbors up to distance k as $BC^k(a) = \bigcup_{b \in R^k(a)} C^k(b)$ and $CC^k(a) = \bigcup_{b \in C^k(a)} R^k(a)$.

Scoring Relationships The main similarity measures used by BC and CC are the *cosine* and *Jaccard* similarity measures. Both measure the relative overlap between two sets A and B but distinguish by the normalization. The cosine similarity is defined as:

$$S_C(A, B) = \frac{|A \cap B|}{\sqrt{|A|.|B|}} \tag{2.1}$$

and the Jaccard similarity as:

$$S_J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.2}$$

Their behavior is relatively the same. They are both equal to 1 for perfect overlap ($A = B$) and equal to 0 for disjoint set ($A \cap B = \emptyset$). They only differ in the numerical value for the intermediate cases. For two BC coupled items a and b considering their indirect references up to distance k, the similarity score using the cosine similarity is measured as $S^k_{BC\text{-}C}(a,b) = S_C\left(R^k(a), R^k(b)\right)$.

2.1.3 Bibliographic Coupling vs Co-Citation Coupling

BC and CC are relatively similar from a graph perspective. BC looks at the references (outgoing edges) while CC looks at the citations (incoming edges). The approaches are precisely the same with the difference that the direction of the edges is reversed for CC. Several works have been done to compare the two approaches [7,8]. In the next paragraphs, we will present some of the differences.

2.1.3.1 Article Decomposition

An article starts with a bibliographic overview, with an *introduction* setting the scope of the field and introducing the problem. Then, *background*, *related works* or *state-of-the-art* sections referencing related methods solving the current problem or a similar one, spot missing points or possible transposition to the current problem that the paper will address. The following part corresponds to the authors' contributions, exposing a novel method supported by an experimental protocol, results, and analysis. The first part presents authors' arguments supported by *bibliographic* evidence, while the second part presents authors' ideas supported by *numerical* evidence.

Most of the time, the article format is imposed on the authors. Therefore, there is a trade-off between the two parts as space is limited. An extensive bibliographic section supports the article with strong argumentation, to the detriment of the main work. On the other hand, a problem introduction that is too short limits the understanding of the working context, the problem and the potential impact. In general, the second part is relatively more extensive than the first, expect for "reviews of" where the aim is to compare the existing literature widely.

2.1.3.2 Evolution over Time

A citation graph is a dynamic structure where new nodes are added over time. The reference count of an article will not change over time (unless the database is consolidated with old documents). However, the number of citations received by an article starts at 0 and will increase over time. This difference between the evolution of $|R(a)|$ and $|C(a)|$ impacts the usage of BC and CC.

An article is bibliographically coupled with other articles at the publication date, unless it has no reference or the quoted papers have never been cited before. Over time, the number of coupled articles will increase as other nodes will refer to its references. The database may not cover very old articles or documents with different types (like patents, blogs, and others). For the nodes referring to these unregistered items, their effective reference set might be incomplete.

In contrast, an article is not co-citation coupled with any other article at the publication date, as no paper could mention this work yet. The citation count will increase over time, increasing the number of coupled items. The coupling strength between two items will vary over time as their respective citation set will evolve. The strength increases if they are mentioned together or decreases if cited separately.

BC is not adapted for very old papers, while CC is not for very recent ones. For not-too-old and not-too-recent papers, they can both be used. BC has the advantage to be usable for freshly published work and even for unpublished works. Therefore, BC is useful to study research fronts or analyze a work under review by looking at its references.

2.1.3.3 Degree asymmetry

The nodes' degree distribution differentiates the evolution of $|BC(a)|$ from $|CC(a)|$. The number of references per article has a low variance, while the variance is much higher for citations. Citations are distributed according to a power-law, where a few works receive many citations and the large majority almost none [9,10]. One theory that could explain this phenomena is the *Matthew effect*, which could be summarized as "the rich get richer and the poor get poorer" [11]. The works that already have many citations will get more than works that have already fewer citations.

The *Matthew effect* tends to model the long-term evolution of $|C(a)|$ as a function of citations' actual number. At the publishing date, all papers are equivalent as they have no citation yet. Different factors impact the initial citation rates of newly published papers. There are *notoriety* factors, such as the publication context (prestigious journals and conferences), the author's affiliation and the author's reputation [11], and *visibility* factors, where the authors could present their work in university, publish a vulgarized version for broader dissemination [12], or write new papers exploiting their work, and open-access [13].

This phenomenon impacts more co-citation analysis than bibliographic coupling as the number of coupled items with BC depends on the references' fame. In contrast, it depends on the paper's citation count for CC coupling which represents a bottleneck.

2.1.3.4 Conceptual Differences between BC and CC

While the approaches are relatively similar, they differ in their interpretation.

Referencing a paper is an *active* action. The authors take their time to look at the literature,

review several articles to select some of them to support their argumentation. The number of references is indirectly limited by the paper format, where an extensive reference list would limit the space available to present the main paper's contribution. Therefore, references must be selected carefully to support the argument concisely.

To be cited is a *passive* process. As a first step, authors can actively contribute to disseminating of their work, by presenting it to their peers, during workshops and other seminars. This would increase the visibility of their work, increasing the likelihood of receiving citations. However, the quantification of all formal and informal interactions the authors had is intractable with the available tools.

Out of the presentation of their work to a targeted audience, the authors have no direct control over who will cite their work. It could be people from the same field, proposing an extension of the current work with new ideas, or people from another field transposing the proposed idea to their problem. Thus, the citation process can be seen as *crowdsourcing*, as an article will be referenced based on its usefulness and not necessarily on the context to which it refers. Therefore, co-citation analysis will give larger scores to items with a similar impact independently of their initial field.

In theory, BC and CC are equivalent for a paper far from the graph boundaries. The number of coupled items can be considered equivalent for both BC and CC if there is a sufficient number of citations and references. It is unlikely that $BC(a)$ and $CC(a)$ broadly overlap because of the graph sparsity and the large number of possible citation choices. Nonetheless, it is a reasonable assumption that they gather items belonging to the same research domain. This might be true most of the time, but a few exceptions exist. One example illustrating the possible asymmetry is *"TensorFlow: a system for large-scale machine learning"* [14]. This paper discusses *distributed computation systems* in its core text and references, which could be used for *Deep Learning*. However, most citations come from Deep Learning papers rather than from distributed systems. This asymmetry could be explained by the largest impact of the technology on *Deep Learning* than on the *distributed system*.

The use of co-citation analysis is more adapted to study the impact of papers, for historiographic purposes. However, this method is not adapted for non-top papers or recent papers as the citation count impacts the coupling highly. In contrast, the bibliographic coupling can deal with high to low quality papers as long as they mention relevant literature, allowing to study scientific communities in their globality and active research fronts. Therefore, we will focus on the BC analysis for the rest of the paper.

2.2 References Selection and its Impact on Coupling Strength

In this section, we will discuss the impact of a reference over the coupling between two papers. First, we will underline the problems using first-order references $R^1(a)$ and high-order references $R^k(a)$. Then, we propose an intermediate solution that considers references' contribution discrediting irrelevant papers to the profit of relevant ones.

2.2.1 Cited and Un-cited Works

Authors have to select a limited set of meaningful references to support their work. We detail in the following some of the reasons governing the choice of a reference rather than another.

Awareness: The first source of omission is the absence of awareness about previous related works. The recentness of a related work limits its propagation through the scientific community. Another reason concerns the vocabulary disparity between two fields. The vocabulary between two field is not necessarily the same; therefore an idea can be described differently. This prevents textual search engines from gathering the two related ideas. The same limitation occurs when papers are published in different languages.

Confidence: The second source of omission is the lack of confidence. ARelevant work may exist, but the authors may not have a sufficient background to understand all the details. The author can also be doubtful about the methods or the obtained results and may prefer not to mention this work.

Notoriety: A factor impacting citation selection is *notoriety*, which contributes to the *Matthew effect*. Out of the fact that top-research or top-journal works are more visible, researchers give them more credit [15]. The best conferences and journals filter papers by expert review in the field only to select best papers. This stamping allows non-expert people to trust the quality of the work by delegating the analysis to a third party.

Knowledge Establishment Major works are cited religiously, like Kerckhoffs [16] for cryptography, or PCA [17] for Mathematics. However, over time, the knowledge is admitted by the field, and there is no need to refer to it, as they tend to become axioms known by everybody. Therefore, the number of citations they would receive per year will vanish over time. The same vanishing process also happens to the theories that are invalidated or outdated by new research. Because those papers are too commonly cited, they are not discriminative enough and increase the computational cost for identifying strongly coupled articles. Therefore, those papers add more processing difficulties with limited benefit.

Independent Discoveries *Multiple independent discoveries* [18] are similar discoveries that occur on a limited time interval, in different places, without interaction between the two (groups of) researchers. The theory behind these events is that these discoveries are the logical

continuum of previous evolution. The findings happened because the current technological advancement made them possible to emerge naturally. As discovery takes time to diffuse and to be accepted, the second group of researchers might not be aware of a previous existence. When two similar works are published, papers following them need to select one or another or both. Whatever the next papers' referencing choice, the two candidates would share the same basis or axioms. At some point, the literature supporting the two works will converge to the same set of concepts and references. Therefore, choosing one or the other does not modify the supporting evidence.

2.2.2 Issues with First-Order References

Bibliographic coupling and co-citation analysis are straightforward conceptual and practical. However, these methods suffer from several drawbacks, which limits their usability. These two approaches explore the first-order references and citations, representing $10 \sim 30$ related works depending on the field, epoch and database.

The main issue concerns the *coupling's strength*. In the case of bibliographic coupling, coupled items have at least one reference in common. Nonetheless, the probability of two coupled papers sharing more than one reference is extremely low because of the sparsity. Thus, coupled strength is more impacted by the total number of references rather than by the number of shared references. Additionally, the similarity between two items sharing 2 or 3 references over 20, i.e. $10 \sim 15\%$ (which is a common value in a scientific citation graph), does not have a great significance.

BC and CC cannot be used to compare randomly selected items as the probability that their reference sets overlap is almost zero due to the graph sparsity. Therefore, it is not possible to identify fields proximity directly with BC.

Different approaches have been put into practice to re-mediate this issue. Some prune papers that are not cited enough [19,20]. These approaches drastically reduce the number of papers, especially recently published papers that did not receive attention yet. Therefore, they are not suitable for the technical watch to study the research fronts [21,22]. Other aggregate papers within a journal [23] or an institution [24,25] level to increase the number of references per entity. The aggregation also decreases of the number of items to analyze, reducing the overall complexity while getting a coarser view.

2.2.3 References' Relevance

A solution to measure papers similarity without merging them in a larger group is to move from first-order neighbors to higher-order neighbors, which must be done carefully.

Assuming a paper makes on average $\mathbb{E}\left[|R(a)|\right] = d$ references, the reference set size when exploring up to distance t can be estimated as $\mathbb{E}\left[|R^t(a)|\right] = d^t$, without counting possible overlap. The relationship can be rewritten differently as $\mathbb{E}[|R^t(a)|] = d \times \mathbb{E}[|R^{t-1}(a)|]$, i.e. the

exploration one step ahead contribute to a multiplication by d of what was here previously. This would lead to a set with mostly old papers, which many other papers would share. Therefore, extending to infinity the exploration distance t will inverse the problem of set overlap to a too large overlap, preventing the identification of meaningful pairs. Additionally, a paper cited as an example coming from another field comes with all its literature independently of its relevance. This mass of irrelevant documents can counterbalance the papers coming from the main field. The last issue is the computational cost, which grows with the set size. To avoid the different issues, the exploration distance t can be selected carefully to limit the set size. However, with a change from d^t to $d^t \times d$, the error can be large making the correct selection of t difficult.

Items in the reference set can be weighted to limit the contribution of irrelevant papers rather than trying to stop graph reference exploration to the correct value t. This weighting must reflect papers' relevance relatively to the initial paper. Relevance can be defined as a function of age where the oldest the document is the less relevant it is. Another way to estimate relevance is to exploit items' reachability from the initial paper. Among the indirectly cited articles are axiomatic papers to which many references are made. Even if the initial paper does not reference them, these papers contributed more significantly to the development of the field of knowledge. In contrast, items that are referenced only once are unlikely to support the paper's ideas strongly. Thus, the weighting would highlight the axioms while naturally ghosting papers with low relevance to the current subject.

The *Random Walk with Restarts* (RWR) is used to weight node contribution for this purpose. This model explores all the different possible paths from papers to referenced papers, putting a larger weight on papers that could be accessed through multiple paths. Therefore, more credit is given to references that are multiple times referred within the reference set, representing a literature consensus. On the other hand, this weighting model discredits papers that are not substantial, like the papers used to illustrate examples but do not contribute to the argumentation, or belong to another field uncommonly cited. Of course, a different weighting model can be proposed as an alternative (considering papers' popularity for instance). Still, the Random Walk is one of the simplest ways to discern *interesting* from *irrelevant* articles [26].

2.3 Measuring Coupling at a Deeper Level

This section presents the different equations allowing us to measure the similarity between two items using the Random Walk with Restarts.

2.3.1 Weighted Similarity Measures

The cosine similarity can be easily extended to weighted elements. Assuming that $\mathbf{W} = \{W_c\}_{c \in A}$ are weights associated with the elements in A and $\mathbf{W}'$ to the elements in B, the cosine similarity can be extended to the weighted domain:

$$S_w(A, B, \mathbf{W}, \mathbf{W}') = \frac{\sum_{c \in A \cap B}(W_c.W'_c)}{\sqrt{\sum_{c \in A}(W_c)^2 \sum_{c \in B}(W'_c)^2}} \tag{2.3}$$

This measure evaluates how well scores attributed in A are similar to scores attributed in B. These similarity measures are equal to 1 if the sets and associated weights perfectly match and 0 if $A \cap B = \emptyset$. This last measure (Eq. (2.3)) will be used to measure the similarity between two papers' backgrounds.

2.3.2 Random Walk with Restarts

For a paper a and a paper $b \in R^t(a)$, the RWR measures the reachability of b starting from a and is denoted W_b^a . The algorithm works by exploring the nodes in $R^t(a)$ starting from a by jumping from one node to another using existing outgoing edges. Explored edges are selected with equiprobability. The exploration "restarts" in a when a path is long enough (here t). The value can be computed recursively starting with $W_a^a = 1$ and 0 otherwise by:

$$W_b^a = \sum_{c \in C(b)} \frac{W_c^a}{|R(c)|} \tag{2.4}$$

Then two papers a and b with respective weights $\mathbf{W}^a = \{W_c^a\}_{c \in R^t(a)}$ and $\mathbf{W}^b$ can be compared by measuring their weighted overlap defined as:

$$S^t_{w\mathrm{BC}}(a, b) = S_w(R^t(a), R^t(b), \mathbf{W}^a, \mathbf{W}^b) \tag{2.5}$$

This coupling is non-zero if a and b share some of their background elements. The strength considers the set overlap and the attributed weights, which can notably change the coupling strength.

2.4 Experimental Setup

2.4.1 Dataset Description

We used for the experiments the DBLP dataset [27] version 12, published in April 2020. It is a research citation graph, composed of 4,894,081 scientific papers with 45,564,149 referencing links. The papers from this database belong to the computer science domain, a small part of the research landscape. An initial processing is done to keep the largest connected component and remove cycles. Cycles exist due to some papers' updates, allowing them to quote papers published after them. For each paper, the database provides information such as title, summary, references, authors, and affiliation, plus around ten descriptive keywords about the domain, field, or methods used.

2.4.2 Node Sampling

We propose to construct a *map of science* exploiting the items' similarity. The computation of a similarity matrix with n elements induces a computational cost of $\mathcal{O}(n^2)$ limiting the usability over very large samples. Random sampling allows to reduce the sampling size easily, and would allow building a *global* map of science. Alternatively, targeted sampling exploiting the metadata available in the dataset allows the construction of a *local* map of science. The keywords associated with papers in the database could be assimilated to a field of study. Therefore, we select one keyword of interest occurring in a sufficient number of papers. All papers associated with the keyword are gathered and then sampled at random to obtain the desired sample size.

2.4.3 Data Visualization

To create *a local map of science*, we used the t-SNE embedding algorithm [28]. This embedding algorithm tries to preserve the neighborhood, i.e., similar items are kept close in the embedding, while distant neighbors are kept away. This algorithm groups similar items, without trying to preserve distances between input and output space. A distance matrix D is given to the t-SNE algorithm, which transforms it into a 2-dimensional representation Y.

The similarity obtained for bibliographic coupling S_{BC} and S_{wBC} have the opposite behavior of a distance, as $S = 1 \iff D = 0$ and $S = 0 \iff \lim_{D\to\infty} D$. It is commonly accepted to transform S into D by $D = 1 - S$. However, the range of possible distance values is restricted to $[0, 1]$, limiting expressiveness. Instead, we propose to transform S into D by inversion:

$$D = \frac{1}{\epsilon + S} - \frac{1}{\epsilon + 1} \tag{2.6}$$

where ϵ is a small constant, set to $\epsilon = 10^{-5}$, which avoids the division by zero cases. ϵ could be considered as the null hypothesis, "what would be the similarity if neighborhoods overlap at random." The pipeline can be summarized as:

$$V \xrightarrow{\text{sampling}} V' \xrightarrow{\text{similarity}} S \xrightarrow{\text{inversion}} D \xrightarrow{t\text{-SNE}} Y$$

2.4.4 Keywords Extraction

The evaluation of embedding Y is non-trivial as visual appreciation is a subjective evaluation. As the proposed methodology is entirely unsupervised and not rely on optimizing a specific criterion, there is no objective value to monitor. Despite this lack of objectivity, we propose to study clusters. The goal is to check if clusters of papers correspond to thematic clusters, described with a particular set of keywords.

Extraction based on Relative Frequency: A possibility to study keywords is to select the most frequent. The keyword used for sampling will be first followed by other keywords

with large predominance. Among those frequent keywords, some are frequent in a general context, and some are frequent only in a specific context. We propose to distinguish between generally frequent keywords from keywords specifically more frequent in the selected domain. To this purpose, we look at the ratio between local and global frequencies, estimated over V' and V. The ratio is defined as:

$$R(kw, V, V') = \frac{r(kw, V')}{r(kw, V)} \tag{2.7}$$

where $r(kw, V)$ represent the frequency of kw within V. A value close to 1 indicates that the keyword is no more frequent in V' than in V, while a larger value indicates that the keyword is specific to this context.

Extraction based on Mean-Shift: The keyword selection process described in 2.4.4 is only possible if the selected sample is relatively singular compared to the whole database. Otherwise, no keyword would be more relevant than another one. Rather than looking at the predominance of a keyword within the subset, its distribution in the embedding can be analyzed to see if the keyword occurs in localized areas or not to evaluate its specificity.

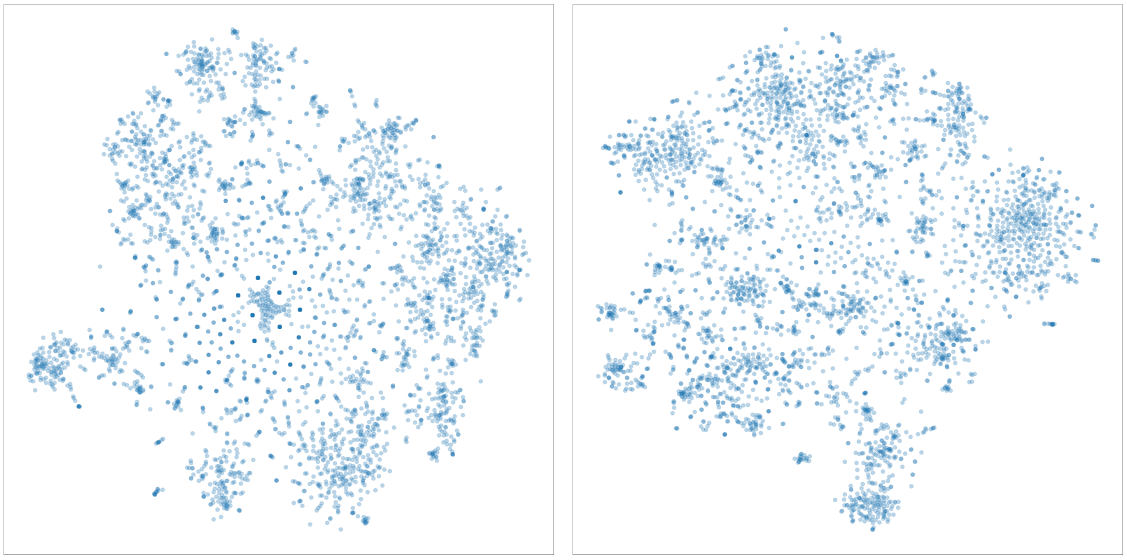
To this aim, the resulting embedding is clustered using the *Mean-Shift* algorithm [29], configured with a Gaussian kernel with bandwidth σ. Then, the embedded items Y are partitioned into k non-overlapping hard clusters $\mathcal{C} = \{C_i\}_{i=1:k}$, where k is automatically found.

Salient keywords are identified for each cluster using TF-IDF. The frequency of a keyword is obtained by measuring the proportion between its number of occurrences against all other keywords within the documents belonging to the cluster.

2.5 Results

2.5.1 t-SNE Embedding: Fields Auto-Organization

A 2D paper visualization is obtained by transforming a similarity matrix using the t-SNE embedding algorithm. For our experiments, we selected 4000 papers associated with the *Payment* keyword. Similar experiments have been run using other keywords, and led to similar results and combination of keywords.



(a) Bibliographic Coupling (BC) (b) Weighted Bibliographic Coupling (WBC)

Figure 2.1: Local Map of Science for "Payment" papers.

Initially, embedding results seem quite similar if we don't consider clusters' location. On both figures, clusters are located on the embedding boundary, while the central areas are less dense with points spaced from each other. Nonetheless, there are main notable differences.

The main one concerns the number of unclustered items, which is larger in the BC figure than in the WBC one. These items are easily identifiable as they are located in the middle and are more distant from their nearest neighbors than items in peripheral clusters. In the case of BC, artefacts are observable forming a symmetrical pattern with a group of items regularly spaced between few dark dots. These dark dots correspond to overlapping papers coupled with the same reference and linked to no other reference. Items at equi-distance from their neighbors are papers coupled with no other document and can be isolated using graph algorithms searching for weakly connected components.

The second major difference concerns cluster delineation. For BC, the cluster density does not vary a lot with the location making difficult the identification of clusters' boundary. For WBC, some boundaries are also difficult to identify. Nevertheless, more clusters show a dense central area and a less dense peripheral area making easier cluster delineation.

These two differences are linked to the high-order neighborhood and the weighting of relevant

references. As a result, the number of isolated items is reduced by the larger number of references, increasing the overall connectivity. In addition, the weighting impacts the formation of dense clusters by having more accurate coupling values leading to a better positioning.

2.5.2 From Keywords to Clusters

2.5.2.1 Relevant Keywords Identification

We compared in these paragraphs keywords ranked by *raw* frequency and ranked by *relative* frequency. For each ranking, we list the 20 first top keywords. Using the *raw* frequency ranking, we have:

> Payment, Computer science, Computer security, Computer network, Mathematical optimization, Economics, Internet privacy, Incentive, Microeconomics, Mathematics, Distributed computing, Marketing, The Internet, Database transaction, Mobile payment

and using the *relative* frequency ratio $R(kw, V, V')$ obtained with Eq. (2.7), keeping only keywords with at least 100 occurrences, we have:

> Microeconomics, Incentive, Database transaction, Mobile payment, Payment service provider, Payment system, Mechanism design, Credit card, Actuarial science, Revenue, E-commerce, Smart card, Cash, Electronic money, Anonymity, Commerce, Common value auction, Cryptocurrency, Electronic cash, Blockchain

When looking at the former list, we can identify some words related to *payment*, like *Economics*, *Microeconomics*, *Marketing* and *Mobile payment*. All others are indirectly related to *payment*, as they correspond to concepts used for the development of different technologies, like *Computer science* or *Mathematics* which are very general concepts and reused in other domains. In contrast, there are much more words related to *payment* in the second list, with only a few words where the direct connection is unclear like *Incentive* or *Actuarial science*. Thus, the proposed ranking allows an uneven extraction of several pertinent keywords without efforts.

2.5.2.2 Keywords Visualization

The keywords in this second list will be studied by highlighting the papers' location where they occurs in the embedding. The goal is to see if groups of papers sharing the same references also share the same keywords.

This measurement cannot be done using only the similarity values. Two coupled papers are unlikely to share more keywords than non-coupled keywords because of the graph sparsity and the low number of keywords per document.

The keywords in the DBLP dataset are organized hierarchically. A document is associated

with general keywords and less general ones. On average, all documents will share the general keywords with their neighbors. When looking at the similarity between two documents' keywords, the contribution of infrequent keywords would be imperceptible.

As the number of keywords per document is limited (on average 10), not all valid keywords are associated with a document. For instance, a *Bitcoin* paper can include *Cryptocurrency* or *Blockchain* keywords. However, its neighbors may be associated with one or none because they are almost equivalent and infrequent. Therefore, the probability that coupled papers would share a specific keyword is very low for infrequent keywords. This difference of predominance makes the direct evaluation of document similarity using keywords problematic.

Among direct neighbors, only a few would share the same specific keywords. Over a larger group, we can expect to gather more papers associated with a particular keyword. The use of an embedding allows considering indirect neighbors, increasing the probability to find neighbors with a particular keyword in common.

In Fig. 2.2, we show keywords location by highlighting papers associated with the current keyword. As a general result, keywords tend to group into clustersis especially true for low-frequency keywords, like *Blockchain* or *Electronic cash*. For keywords with larger frequency, like *Database transaction*, *Payment service provider*, or *Microeconomics*, they occur in multiple clusters, possibly disjoint. Finally, some keywords do not aggregate into dense clusters, such as *Commerce*, *E-commerce* or *Revenue* which form diffused areas, partly due to their low frequency compared to other keywords. Low-frequency keywords correspond to very specific topics, with a narrowed literature. The ability to group and form a dense cluster is more likely for specific topics than for more general topics. A broad topic can be sub-divided into subtopics. A paper within a subtopic may have general references relative to the broad topic but others related to the subtopic. Papers from different subtopics are likely to be weakly coupled, leading to the decomposition of a large topic into multiple clusters.

This map can also be used to study keywords relatedness by looking at their covered areas. We can identify several groups of related keywords, like *Blockchain* with *Cryptocurrency*; *Electronic money* with *Electronic cash*, *Cash* and *Anonimity*; *Common value auction* with *Mechanism design*; *Credit card* with *Smart card*, etc. Some keywords share exactly the same areas like *Blockchain* and *Cryptocurrency*, while others partially overlap, like *Anonymity* overlapping with *Blockchain* in the one side and *Electronic cash* and related keywords for the other side. The partial overlap shows that concepts are not hierarchically organized, as none dominate the other. Nevertheless, a research area can be charaterized by a specific mixture of topics.

Figure 2.2: Local Map of Science for "Payment" papers with papers associated with a relevant keyword highlighted. Keywords are arranged based on their co-location similarity.

2.5.3 From Clusters to Keywords

In this experiment, we start from the papers' clusters to study keywords' groups instead of starting from keywords' clusters to identify papers' groups.

2.5.3.1 Mean-Shift Partitioning

The WBC embedding is clustered using the Mean-Shift algorithm, where $\sigma = 2.8$, corresponding to the average distance to the 30 first nearest neighbors. The partitioning results are presented in Fig. 2.3, where each color is associated with a cluster. The clustering led to the identification of 38 clusters, of which 34 have at least 30 documents.

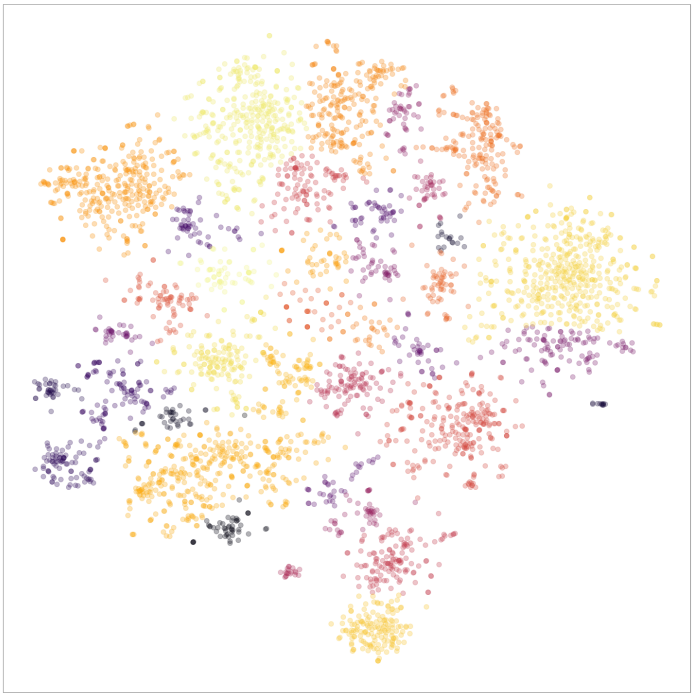


Figure 2.3: Mean-Shift clustering of the *Payment* resulting embedding.

The partitioning obtained led to the identification of clusters of various size and shape. Compared to a k-Mean, the Mean-Shift algorithm has the advantage of not requiring as input the number of clusters and separating clusters by non-straight boundaries. However, some potential "errors" exist, like the two clusters on the right (a yellow and a purple) would be better merged. In the middle of the embedding, different clusters are identified. As these documents correspond to weakly coupled items, their relevance and usability are unclear.

2.5.3.2 Group Analysis

Keywords are analyzed by ranking all keywords within a cluster using TF-IDF. We assigned a keyword to a cluster based on its largest TF-IDF score to avoid multiple occurrences. Some groups are presented in table 2.1, where keywords within a group are listed by decreasing score and keeping only those with several occurrences within the subset equal or greater than

20. We kept the top-most relevant keywords as a matter of space. Among the 34 cluster candidates, we selected 28 of them, the clusters containing the previously identified keywords and others with a sufficient number of relevant keywords for comparison.

Among the 20 keywords in the second list, only *Mechanism design* is not listed because of its poor TF-IDF score. Only a minority of the keywords in the first list is listed, as their large number of occurrences leads to a very low IDF score.

The topic covered by each cluster is easy to identify. Often, one of the keywords within the group is a good representative. For example, for group 1, *Cryptocurrency* represents the main idea. For group 11, the *Biometry* topic includes all the other sub-topics. Group 16 with *Mathematical economics*. For other groups, the idea can be summarized by using an external keyword. For instance, group 3 corresponds to payment technologies, including protocols and devices. For group 20, the idea can be summarized by user-related security.

There are some groups where the concept is unknown, such as group 6 or group 27, which is partially due to the clustering and the single assignment of a keyword. In addition, there are some keywords where the links to the group idea are unclear, such as *Fuzzy logic* in group 25, or *Digital goods* in group 2.

Words may have multiple meanings or cover different issues. Within a group, the correct meaning becomes evidence. For instance, *Database transaction* refers to many things. It could be the algorithms, ensuring ACID principles (Atomicity, consistency, isolation and durability), or refers to the recorded data. Within the *Cryptocurrency* cluster, the meaning becomes very clear, as one of the main blockchain issues is how to record the maximal number of transactions in a decentralized system without sacrificing security and correctness. Associated with another cluster, the interpretation would have been different. We assign each keyword to a cluster only once to avoid very long keywords lists; therefore we don't have another contextual example. Nevertheless, for a more in-depth analysis, all relevant keywords must be considered.

This experiment allows identifying clusters' topics based on the main keywords. Another experiment would be to study topics relationships by looking at the nearest clusters mixture. Nonetheless, this approach is not recommended over a t-SNE embedding. This embedding method tries to preserve the local neighborhood, and distant neighbors are not organized specifically. Therefore, conclusions concerning clusters' relationships might be inexact.

Table 2.1: Table gathering keywords grouped by clusters and sorted by TF-IDF.

Group 1	Group 2	Group 3	Group 4
Cryptocurrency	**Electronic cash**	**Mobile payment**	Micropayment
Blockchain	**Anonymity**	**Payment service provider**	**Electronic money**
Ledger	Blind signature	Payment gateway	Cryptographic protocol
Smart contract	Trusted third party	Mobile commerce	Hash chain
Communication channel	Computer security	**Credit card**	**E-commerce**
Scalability	Money laundering	Payment protocol	Hash function
Throughput	Digital goods	Card security code	3-D Secure
Currency		Digital signature	Payment order
Database transaction			

Group 5	Group 6	Group 7	Group 8
Near field communication	Multimedia	Virtual machine	Demand response
Smart card	Digital currency	Provisioning	Electricity
Access control	Welfare	Cloud computing	Smart grid
Contactless smart card	**Cash**	Resource management	Control engineering
ATM card	Social media	Cost accounting	Environmental economics
	Financial management	Resource allocation	Energy consumption

Group 9	Group 10	Group 11	Group 12
Ubiquitous computing	Network packet	Biometrics	Technical debt
Mobile device	Wireless ad hoc network	Fingerprint	Systems engineering
Mobile computing	Wireless network	Authentication	Software
Embedded system	Relay	Password	Process management
Payment system	Mobile telephony	Usability	Risk analysis
World Wide Web	Security token		Management science
Commerce	**Incentive**		Empirical research

Group 13	Group 14	Group 15	Group 16
Dividend	Operation management	Online advertising	Mathematical economics
Econometrics	Supply chain	**Common value auction**	Vickrey-Clarke-Groves auction
Optimization problem	**Microeconomics**	Advertising	Combinatorial auction
Actuarial science	Economics	Bidding	Redistribution
Interest rate	Information asymetry	Valuation	Optimal mechanism
Transaction cost	Profit	**Revenue**	Social Welfare
Mathematics	Industrial organization	Budget constraint	

Group 17	Group 18	Group 19	Group 20
Net neutrality	Cognitive radio	Ransomware	Android
Bargaining problem	Price of anarchy	Malware	Internet privacy
Quality of service	Cellular network	Encryption	Security analysis
Internet access	Stackelberg competition	Monetization	Information sensitivity
Game theory	Operator	Payment processor	Implementation
Profitability index	Wireless		Vulnerability
Nash equilibrium	Telecommunications network		Information security

Group 21	Group 22	Group 23	Group 24
Welfare economics	Knowledge management	Crowdsourcing	Decision tree
Linear programming	Medicine	Data science	Transaction data
Schedule	Information technology	Transparency	Artificial neural networks
Public transport	Outsourcing	Remuneration	Machine learning
Heuristics	Decision support		Artificial intelligence
Exploit			Big data

Group 25	Group 26	Group 27	Group 28
Trade credit	Technology acceptance model	Crowdsensing	Peer-to-peer
Economic order quantity	Risk perception	Rationality	Upload
Inventory control	Marketing	Reverse auction	Game theoretic
Inventory shortage	Psychology	Computation	Broadcasting
Inflation	Perception		Popularity
Mathematical optimization	Developing country		Collusion
Vendor			Authorization
Fuzzy logic			

2.6 Discussion

2.6.1 Would Bibliographic Coupling be Sufficient ?

The main issue with BC is the lack of connectivity between papers, leading mechanically to low similarity scores, and few coupling relationships. Our experiment presented a subset of a paper belonging to the *Payment* domain, a small world with many connections between sub-topics. In the broader area, such as *Machine Learning*, *Computer security*, *Information retrieval*, there can be more than 100,000 documents. When randomly sampling papers from these domains, the probability of papers sharin references is low leading to many more artefacts than a narrowed field.

2.6.2 Calibration of Random-Walk Path Length

The WBC exploits weights computed using the random walk with restart assigned to $R^t(a)$ items. The setup with $t = 1$ corresponds to BC as only the first neighbors are explored with equal probability. The value of t is adjusted with the sparsity of the subset. A very dense group of items requires a low value of t, around 2 or 3, while a very sparse structure needs a larger value of $4 \sim 5$ However, a larger t impacts the computational cost; therefore using the lowest possible value is preferable. We set $t = 3$ for the experiments as it allowed us to test over sparse and less sparse subsets.

2.6.3 Node Sampling

We proposed a methodology for representing papers without prior knowledge. The papers described here belong to the same subset. Still, they are selected at random, without looking at age, number of received citations, number of references, language, conference or journal ranking. The only bias was on the thematic, which narrows the scope and gives understandable relationships. Other selection approaches can be proposed, such as looking at all papers published at a large conference or journal, for automatic field categorization. Another would be to look at a university's papers, to study synergies between different research groups.

2.6.4 Bottleneck Effect

The proposed approach takes advantage of the random walk, where the influence of a node diffuse over the graph. At the start, a weight of 1 is dispatched between all first neighbors. With a larger exploration distance, the amount dispatched is still 1, or lower if some terminal nodes are reached. Therefore, the system is almost conservative. The amount gathered by a node depends on the distance. For a low exploration distance, the weights gathered vanished because they are dispatched over many nodes. After more exploration steps, the number of beneficiaries decreases because the set of papers in the past is smaller than in the recent years. Therefore, old papers tend to concentrate the amount of diffused weights surpassing the amount collected by recent works. The bottleneck effect can be avoided by slightly

modifying the random walk by adding a decay factor α, to consider the loss or transformation of information over time.

2.6.5 Future versus Past

As discussed earlier, an asymmetry exists between citation and referencing process. Nonetheless, the proposed approach can be used over citations rather than references, or a mix of the two. The referencing approach focuses on "who share the same basis," while with the citation approach, the focus would be "who are co-inspired." The conceptual difference might be relatively small, but more artefacts are likely to be present because of the power-law distribution of citations, as many papers received very few citations. However, this way might be a useful exploration path for papers impact evaluation.

2.7 Conclusion

In this paper, we proposed a methodology to extend bibliographic coupling to the more in-depth neighborhood. The presented approach considers the indirect neighborhood of a specific paper, where their contribution is weighted using the Random Walk algorithm. This approach allows to weight neighborhood nodes based on their relative distance and accessibility from the parent paper, which corresponds to a form of relevance. These weights are used to compute the similarity between two papers, using a weighted Cosine similarity.

The advantage against bibliographic coupling is the connectivity: much more similarity pairs are non-zero, which create a single connected component. Following an embedding step with the t-SNE algorithm contains much fewer artefacts than bibliographic coupling, which leads to a better analysis of the sub-field relationships. Because of the connectivity advantage, any papers could be represented on our map, without restriction on the number of citations. This enlarges the possible candidate papers which can be used to build local maps of science.

CHAPTER 3

Index t-SNE: Tracking Dynamics of High-Dimensional Datasets with Coherent Embeddings.

t-SNE is an embedding method that the data science community has widely used. Two interesting characteristics of t-SNE are the structure preservation property and the answer to the crowding problem, where all neighbors in high dimensional space cannot be represented correctly in low dimensional space. t-SNE preserves the local neighborhood, and similar items are nicely spaced by adjusting to the local density. These two characteristics produce a meaningful representation, where the cluster area is proportional to its size in number, and relationships between clusters are materialized by closeness on the embedding.

This algorithm is non-parametric, therefore two initializations of the algorithm would lead to two different embedding. In a forensic approach, analysts would like to compare two or more datasets using their embedding. An approach would be to learn a parametric model over an embedding built with a subset of data. While this approach is highly scalable, points could be mapped at the same exact position, making them indistinguishable. This type of model would be unable to adapt to new outliers nor concept drift.

This paper presents a methodology to reuse an embedding to create a new one, where cluster positions are preserved. The optimization process minimizes two costs, one relative to the embedding shape and the second relative to the support embedding' match. The proposed algorithm has the same complexity than the original t-SNE to embed new items, and a lower one when considering the embedding of a dataset sliced into sub-pieces. The method showed promising results on a real-world dataset, allowing to observe the birth, evolution and death of clusters. The proposed approach facilitates identifying significant trends and changes, which empowers the monitoring high dimensional datasets' dynamics.

3.1 Introduction

High dimensional datasets are very rich sources of information. Because of their wideness, they are difficult to investigate, process, and represent in a simple manner.

An appropriate reduction of width would benefit from:

- Storage cost reduction;
- Computational cost reduction;
- Denoising / Information compression;
- Synthetic data visualization.

For a dataset $X \in \mathbb{R}^{n \times d_0}$, with n elements and d_0 dimensions, a reduction into $Y \in \mathbb{R}^{n \times d_1}$ offers a reduction of the memory footprint of $100(1 - \frac{d_1}{d_0})$ %, which is non-negligible for large datasets. For linear algorithms, the computational cost is reduced by the same factor.

The core information in those wide datasets is hard to identify because of the large number of features and correlation, and redundancy. The use of methods that condense the information enables to obtain a synthetic view of the dataset, which would benefit post-processing algorithms or data analysts' work. The synthetic view can also be used to display results by coloring items according to their predicted value.

A dimension reduction can be performed following different approaches:

- Automatic feature selection;
- Human engineered feature;
- Automatic feature extraction;

Automatic feature selection selects some features among all available according to some characteristics. The feature importance can be evaluated using Shapley values [30] or removing redundant features [31]. These approaches are relatively straightforward to put into practice. Nonetheless, some information may be lost if the signal is too weak, or if the number of selected features is too small.

Feature engineering allows shaping human knowledge into an algorithmic form. This approach is practical when a minimal amount of data is available, preventing the use of automatic algorithms. It is necessary in specific cases to transform human data into processable information. Dates are a good example: a computer cannot understand directly that $\mathbf{x} = [28, 2]$ and $\mathbf{y} = [1, 3]$ represent dates and that $\|\mathbf{x} - \mathbf{y}\| = 1$ or $2\}$ days, depending on this is a leap year or not. Apart from this example with an exact answer, there is no guarantee if the transformation would help or prevent later algorithms from performing good predictions. As these features are not learned from data, they are likely to be stable, ie not tricked by outliers which could perturb the learning. They offer some form of *explainability*, as the engineer can describe the meaning of the transformation in the human language. In the

absence of previous knowledge, their development is costly in operating time, and on a large dataset, an expert may miss some essential features.

Automatic feature extraction replaces handcrafted features with algorithm-learned features. This is a broad research area, with statistical compression methods such as Principal Component Analysis [17], or methods based on deep neural networks, like autoencoders [32], [33].

Automatic feature extraction can be decomposed into two categories, based on reusability on new data:

- Parametric methods;
- Non-parametric methods.

Parametric methods, such as PCA [17], Self-Organizing Maps [34], learn a mapping function $f : X \to Y$, which minimize a quantity of the form $\arg_f \min Cost(X, f(X))$. When learned, the function f can be reused on any new input X'. The inference time of most of these methods is linear, such as $f(X) = [f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_n)]$. For a large input X, the computation can be distributed on several machines, which enhance scalability to large datasets.

In contrast, non-parametric methods minimize a specific quantity $\min_Y Cost(X, Y)$ directly by optimizing Y's values. No function is learned, which prevents the reusability of a previous computation for new data. This is the case of ISOMAP [35], UMAP [36] or t-SNE [28]. Despite the non-reusability, the two most recent methods, t-SNE and UMAP, have been extensively used by the machine learning community. Those methods have been successful at representing high dimensional datasets, by adapting to disparate scaling and non-homogeneous densities while letting appear clustering structures.

The strength of t-SNE comes from its ability to deal with heterogeneous scaling and the crowding problem. In high dimensional space, an item may have many neighbors around, all distant from each other. By reducing the number of dimensions, it is impossible to preserve the distance between an item and its neighbors and between the neighbors. If distance to the item is preserved, neighbours' distance will decrease, making them closer than in high dimensional space. This corresponds to the crowding problem.

Instead of preserving all the distances, t-SNE preserves it locally. The algorithm adapts for each item to the local density, taking into account a small group of neighbors. This local adaptation makes the power of t-SNE, as the points in a very dense cluster are separated from each other. Consequently, the number of items in a cluster is proportional to its visual area on the embedding, which helps an analyst look at the dataset composition. The other impact of the local adaptation is on the ability to deal with heterogeneous data scaling. All items fit together on the same visual space regardless their initial distance to the dataset's mean. These characteristics lead to embeddings with excellent visual qualities.

Despite the high quality of the obtained embeddings, t-SNE is a non-parametric method,

where no function is learned. The outcome of running t-SNE twice with the same input X leads to two different embeddings $Y^{(0)}$ and $Y^{(1)}$, with no equivalence between the positions. This is due to the initialization, which starts with randomly generated vectors.

The initialisation process can be controlled to improve determinism. Many works such as [37] proposed to initialize the embedding with PCA coefficients. Starting with these positions improves the repeatability, but does not ensure the regeneration of large scale structures for different datasets, as t-SNE preserves local neighborhood only. The work of [37] proposes a method to create large scale structures using two t-SNE steps, which enables to obtain embeddings with large and low scale similarities. The algorithm starts with the PCA coefficients as initial item positions, followed by a t-SNE step adjusted to take into account far-range neighborhood. These positions are reused by another t-SNE step, taking into account small-range neighborhood, letting appear a finer structure. This work was successful for visual analytics, allowing the preservation of relative cluster positions over multiple datasets. The embeddings are visually similar, but the preservation of cluster positions is not exact, preventing the use of the same algorithm on all embeddings.

A naive approach to compare two datasets using their embedding is to compute the joint embedding over the consolidated dataset $X'' = [X, X']$. There are two limitations to this approach. The first one concerns the computational cost, as the complexity of t-SNE is in $\mathcal{O}(n^2)$ for the worst case. The work of [38] proposes an approximation of the different forces, claiming a linear complexity. Nonetheless, the second limitation concerns the data availability. If the two datasets are available now, but a third would arrive later, the embedding corresponding to $[X^{(0)}, X^{(1)}]$ would not share spatial correspondences with the embedding obtained with $[X^{(1)}, X^{(2)}]$.

To obtain consistency in the item positioning, several works [39,40] proposed the use of deep neural networks to mimic the behavior of t-SNE. As with any trained algorithm with no memory nor update mechanism, the inference results is purely deterministic. The algorithm would be able to map correctly a dataset with a distribution similar to the training dataset However, for a dataset with a different distribution, the model would not adapt to the new density, leading to overcrowded and/or depleted areas. The neural network would not be able to adapt to concept drift, nor as new outliers as these models' generalizability is limited to their training set.

Instead of learning how to make an embedding, LION t-SNE [41] proposed an answer to *where new points should be put*, taking into account the points already present and the empty areas left. New points are positioned nearby their nearest neighbors without moving existing items from their location. Items that do not have relevant neighbors in the input space are positioned on an empty area of the embedding, filled later with more relevant neighbors. This approach allows adding a few points on the previous embedding, keeping the possible visual quality of the embedding. While this work deals correctly with item's addition, normal or outlier, it does not deal with update nor deletion. Last point concerns the scalability. The

addition of a few points is likely to preserve the general aspect of the embedding. However, the shape of the resulting embedding after a massive addition of items is unknown.

Last work to mention is Dynamic t-SNE [42] which updates the embedding $Y^{(t)}$ into $Y^{(t+1)}$ after a change from $X^{(t)}$ into $X^{(t+1)}$. It assumes that there is a one-to-one correspondence between items of $X^{(t)}$ and $X^{(t+1)}$. This setup corresponds to a monitoring situation, where the data coming from a fixed number of sensors arrives at each time step. To compute $Y^{(t+1)}$, *dt*-SNE starts with the previous embedding positions $Y^{(t)}$, and tries to minimize the cost defined by *t*-SNE relatively to $X^{(t+1)}$. A penalty is added on the displacement of $Y^{(t+1)}$ from the initial position, which enables to keep the embedding coherent over multiple time steps. While this work addresses the updatability, as no items can be added nor deleted, the usability is restricted to particular use-cases, such as multidimensional time series.

In this paper, we abord the problem of the reusability of a *t*-SNE embedding. The proposed approach is inspired by *dt*-SNE, concerning the idea of using the previous embedding as a support to the new embedding. The support embedding is not used to initialize the new item positions but to guide them towards neighbors location. Compared to *dt*-SNE, the scope is broadened because there is no constraint on the integrated elements, nor on their number or distribution. By enlarging to the addition, updating and deleting items, our method can be used in many more real-world monitoring situations, such as when some sensors are added to the system or removed due to failure. The approach is not limited to the temporal dataset, but to any *index* variable, a discrete or continuous variable, such as temperature or speed. Dataset can be sorted according to this variable, and successive embeddings can be issued to track the impact of the *index* variable over the data distribution. In other words, it allows to obtain embedding conditional to the index variable of interest. The method is called *index* *t*-SNE, abbreviated *it*-SNE, for this reason.

In the first section, the main equations governing *t*-SNE optimization process are introduced. This section is followed by the description of *it*-SNE, reusing part of the initial *t*-SNE scheme. Then, the methods section describe the different datasets and evaluation metrics, followed by the experimental results. Last, this article finishes with a discussion followed by its conclusion.

3.2 t-SNE Formulation

t-SNE [28] is a structure-preserving embedding algorithm trying to preserve the local neighborhood of items in a low dimensional space. Given a dataset $X \in \mathbb{R}^{(n \times d)}$, of n items lying in a d dimensional space, the goal is to generate its corresponding embedding $Y \in \mathbb{R}^{(n \times d_e)}$:

$$Y \leftarrow t\text{-SNE}(X; d_e, perp)$$

where d_e is the number of embedding dimensions, often set to 2, and *perp* is the perplexity parameter. Two items i and j neighbors in X must be neighbors in Y. The definition of *neighbors* depends of two things: the local density around an item, and the user-defined perplexity parameter which represents the average number of neighbors to consider. Rather than reasoning in terms of distances, the algorithm uses probabilities, computed from pairwise distances, to optimize Y.

3.2.1 Interaction Probability

t-SNE tries to adapt the embedding vector Y to X using their respective probability matrices P and Q, both of dimension $n \times n$. These probabilities represent the degree of relatedness of two items in their respective space. A large probability correspond to a high proximity, while a smaller to a large distance. The input and output probability matrices are computed differently to create a small asymmetry.

3.2.1.1 Input Probability Matrix

The conditional probability of item j with respect to i is defined as:

$$p_{j|i} = \frac{1}{Z_i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right) \tag{3.1}$$

By convention, $p_{i|i} = 0$ as an item does not interact with itself, and $Z_i = \sum_{j \neq i} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right)$ is the normalization constant of item i, such as $\sum_j p_{j|i} = 1$.

The standard deviation parameter σ_i adapts the kernel range to the local density around item i. The optimal value of σ_i is obtained by binary search to match the *perplexity*. The perplexity is a user-defined parameter that represents the average number of neighbors of an item, defined formally as:

$$Perp(P_i) = 2^{H(P_i)} \tag{3.2}$$

where $H(P_i)$ is the Shannon entropy calculated as:

$$H(P_i) = -\sum_{j \neq i} p_{j|i} \log_2(p_{j|i}) \tag{3.3}$$

The joint probability between i and j is defined as $p_{i,j} = \frac{p_{i|j} + p_{j|i}}{2n}$. These equations enable the computation of the symmetric probability matrix P given a particular dataset X and a perplexity target.

3.2.1.2 Output Probability Matrix

The output probability matrix Q is obtained in a similar manner using the embedding vector Y. As the goal is to obtain homogeneous distances between neighbors, there is no adaptation to local neighborhood. Another difference concerns the kernel choice. Instead of an exponential kernel, a t-student kernel with one degree of freedom is used. This kernel asymmetry allows modifying the long-range interactions, which leads to repulsive forces between non-neighbors items.

The joint probability between item i and j is calculated as:

$$q_{i,j} = \frac{1}{V}(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1} \tag{3.4}$$

with $q_{i,i} = 0$ and $V = \sum_{k \neq \ell}(1 + \|\mathbf{y}_k - \mathbf{y}_\ell\|^2)^{-1}$ the global normalization constant, which lead to $\sum_{i,j} q_{i,j} = 1$.

3.2.2 Cost Minimization

The dissimilarity between the two probability matrices P and Q is measured using the Kullback-Leibler divergence:

$$KL(P\|Q) = \sum_i C_i = \sum_i \sum_{j \neq i} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} \tag{3.5}$$

where C_i the cost associated to item i.

By deriving equation (3.5), a simple form of the gradient is obtained:

$$\frac{\partial C_i}{\partial y_i} = 4 \sum_j (p_{i,j} - q_{i,j}) \frac{\mathbf{y}_i - \mathbf{y}_j}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2} \tag{3.6}$$

The algorithm uses the gradient descent approach to minimize the cost by updating the initial $Y(0)$ solution:

$$\begin{aligned} Y(t) = \quad & Y(t-1) - \alpha(t) \tfrac{\partial C}{\partial Y}(t) \\ & + \eta(t)\,(Y(t-1) - Y(t-2)) \end{aligned} \tag{3.7}$$

with $\alpha(t)$ the learning rate, adjusted over time, and $\eta(t)$ the momentum rate. The matrix Q is recomputed at each update step according to the newly obtained $Y(t)$, while the matrix P is not as the input vector is left unchanged. The computation of Q at each step is the most costly operation, which leads to a complexity in $\mathcal{O}(n^2)$ without optimization.

In the original paper [28], many optimization tricks are used. For instance, *early exaggeration* replaces temporary $(p_{i,j} - q_{i,j})$ by $(kp_{i,j} - q_{i,j})$, where $k > 1$. This trick amplifies the attraction forces between nearest neighbors, which fasten the formation of separated clusters. After some steps, the factor is set back to $k = 1$ which lets nearest neighbors to separate from each others.

Another trick is to add gaussian noise of small amplitude to the gradient to get out of local minima at start.

The last point to be made is the $\alpha(t)$ learning rate. This rate is updated at each step to boost it in the right directions and slow it down in uncertain situations.

3.3 Indexed t-SNE

The initial formulation of t-SNE leads to a different embedding for each new initialization. Instead of learning a new embedding from scratch, *it*-SNE takes advantage of prior embedding to optimize the new embedding.

Given a support dataset $X^{(0)} \in \mathbb{R}^{n_0 \times d}$ of n_0 items and its corresponding embedding $Y^{(0)} \in \mathbb{R}^{n_0 \times d_e}$, the goal is to generate an embedding $Y^{(1)} \in \mathbb{R}^{n_1 \times d_e}$ corresponding to $X^{(1)} \in \mathbb{R}^{n_1 \times d}$ of n_1 items.

$$Y^{(1)} \leftarrow \textit{it}\text{-SNE}(X^{(1)}, X^{(0)}, Y^{(0)}; perp) \tag{3.8}$$

Two items i and j neighbors in the input space must be neighbors in the embedding space, regardeless of their origin dataset.

3.3.1 Cost

To achieve this goal, *it*-SNE minimizes two independent costs:

- the *intra* cost, $C^{(1)}$, defined as in t-SNE using $(X^{(1)}, Y^{(1)})$;
- the *inter* cost, $C^{(0,1)}$, corresponding to joint interactions between $(X^{(0)}, Y^{(0)})$ and $(X^{(1)}, Y^{(1)})$.

The total cost to minimize is:

$$C^{(1)}_{tot} = C^{(1)} + C^{(0,1)} \tag{3.9}$$

The formulation of $C^{(1)}$ is unchanged from the initial algorithm formulation.

3.3.2 Interaction Probability

Similarly to t-SNE, the probability matrices are defined to represent items relationships. $P^{(0,1)}$ denotes the probability matrix between input data $X^{(0)}$ and $X^{(1)}$, and $Q^{(0,1)}$ for their respective embeddings $Y^{(0)}$ and $Y^{(1)}$.

3.3.2.1 Input Interaction

For two items $\mathbf{x}_i \in X^{(0)}$ and $\mathbf{x}_j \in X^{(1)}$, the input probability is defined as:

$$p^{(0,1)}_{i|j} = \frac{1}{Z'_j} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_j^2}\right) \tag{3.10}$$

where σ_j corresponds to the optimal parameter for j obtained with t-SNE on $X^{(1)}$, and $Z'_j = \sum_i \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_j^2}\right)$ is the normalization constant to obtain $\sum_i p^{(0,1)}_{i|j} = 1$.

The joint probability between i and j is defined as:

$$p_{i,j}^{(0,1)} = p_{j,i}^{(1,0)} = \frac{1}{2}\left(\frac{p_{i|j}^{(0,1)}}{n_1} + \frac{p_{j|i}^{(1,0)}}{n_0}\right) \tag{3.11}$$

The symmetrization allows taking into account the density of the two datasets on a given location. Additionaly, the normalization allows to equalize the dataset influence. If one dataset is larger than the other, it would contribute more are the total forces from each of its items would be larger than for the smaller dataset. The normalization by dataset size allow to obtain equivalent contribution.

3.3.2.2 Output Interaction Probabilities

The goal of *it*-SNE is not to place new items $Y^{(1)}$ on existing holes of $Y^{(0)}$, but to have $Y^{(1)}$ on a parallel layer of $Y^{(0)}$. To relax forces, and to take into account embedding separation, a penalty factor ϵ is introduced to artificialy separate points belonging to different embeddings. It could be seen as new embedding dimension $d_e + 1$, such as $\mathbf{y}_i^{(0)} = [y_{i,1}^{(0)}, y_{i,2}^{(0)}, ..., y_{i,d_e}^{(0)}, 0]$ and $\mathbf{y}_j^{(1)} = [y_{j,1}^{(1)}, y_{j,2}^{(1)}, ..., y_{j,d_e}^{(1)}, \epsilon]$. The distance between two items is then $\|\mathbf{y}_i^{(0)} - \mathbf{y}_j^{(1)}\| = \|\mathbf{y}_i - \mathbf{y}_j\| + \epsilon^2$.

The kernel used for the definition of output probabilities is kept unchanged, up to the addition of ϵ:

$$q_{i,j}^{(0,1)} = \frac{1}{V'}\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2 + \epsilon^2\right)^{-1} \tag{3.12}$$

where $V' = \sum_{i,j}\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2 + \epsilon^2\right)^{-1}$ is the normalization constant, which ensures $\sum_{i,j} q_{i,j}^{(0,1)} = 1$.

3.3.2.3 Cost Minimization

The modification of *t*-SNE algorithm has a limited impact on the cost derivative formulation. The only change impacts the strength of the gradient by the addition of the term ϵ^2:

$$\frac{\partial C^{(0,1)}}{\partial \mathbf{y}_i^{(1)}} = 4\sum_j (p_{i,j}^{(0,1)} - q_{i,j}^{(0,1)}) \frac{\mathbf{y}_j - \mathbf{y}_i}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2 + \epsilon^2} \tag{3.13}$$

The initial solution $Y^{(1)}(0)$ is optimized following equation (3.7), replacing $\frac{\partial C}{\partial Y}$ by $\frac{\partial C^{(1)}}{\partial Y^{(1)}} + \frac{\partial C^{(0,1)}}{\partial Y^{(1)}}$.

3.4 Experimental Setup

3.4.1 Algorithm Parametrization

For all experiments, the target perplexity is set to 30. The initial vector of Y, used to start the optimization process is drawn from a gaussian distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma = 10^{-4}$. At each iteration step, a gaussian noise of standard deviation $\sigma = 10^{-4}$ is added to the gradient.

The initial learning rate $\alpha(0) = 10$ is adapted at each timestep for each item i and dimension k according to the similarity between the gradient and the previous displacement direction $\delta_{i,k}(t) = -\frac{\partial C}{\partial y_{i,k}}(t).(y_{i,k}(t-1) - y_{i,k}(t-2))$:

$$\alpha_{i,k}(t) = \begin{cases} \alpha_{i,k}(t-1) + 0.2 & \text{if} \quad \delta_{i,k}(t) > 0 \\ \alpha_{i,k}(t-1) \times 0.8 & \text{else.} \end{cases}$$

The momentum rate $\eta(t)$ is adapted over learning, such as $\eta(t) = 0.5$ if $t < 250$ else 0.8.

For t-SNE, the early exaggeration trick is used for the 100 first steps, with an exaggeration factor of 2. For it-SNE, the early exaggeration was disabled.

The number of training steps for a t-SNE embedding and it-SNE is fixed to 300 and 200 respectively. Unless specified, $\epsilon = 1$.

t-SNE and it-SNE were implemented in Python, using NumPy library [43].

3.4.2 Datasets

We propose to illustrate the results it-SNE over two types of datasets:

- A synthetic dataset, where all parameters can be controlled at ease;
- A real-world dataset to look at its capabilities on unknown distributions.

3.4.2.1 High Dimensional Gaussians

High dimensional Gaussians are good study candidates, as all dimension are equivalent, preventing the use of renormalization methods. The dataset was constructed inspired from the protocol described in [42].

A dataset of 100 dimensions is created by generating gaussian clusters. 10 gaussian center positions $\{\boldsymbol{\mu}_g\}_{g=1:10}$ are generated uniformly at random in $\boldsymbol{\mu}_g \in [-0.5, 0.5]^{100}$. For each gaussian g, 100 items are sampled from the multivariate normal distribution of mean $\boldsymbol{\mu}_g$ and standard deviation σ. This process leads to a dataset of 1000 items.

In the paper Dynamic t-SNE [42], the authors proposed to build a temporal dataset made of shrinking Gaussians. For each item $\mathbf{x}$ associated with the Gaussian g, the distance between the item and the center is reduced by 10 % at each time-step, such as $\|\mathbf{x}(t) - \boldsymbol{\mu}_g\| = 0.9^t \|\mathbf{x}(0) - \boldsymbol{\mu}_g\|$. Mechanically, $\sigma(t)$ is reduced in the same proportion, such as $\sigma(t) = 0.9^t \sigma(0)$. For this

experiment, $\sigma(0) = 1.0$. The shrinking process is followed for 9 steps, which leads to $\sigma(10) \approx 0.4$.

By changing the time direction, a similar experiment with growing Gaussians is generated, starting with $\sigma(0) = 0.4$, and growing at the rate of $\sigma(t) = 0.9^{-t}\sigma(0)$.

While these two introductive experiments keep the total number of point stable and homogeneous density for each Gaussian, we propose to build a temporal dataset where heterogeneity appears over time. For each Gaussian g, an expansion rate is sampled uniformly at random from $r_g \in [0.5, 1.5]$, where 1. leads to an invariance of the element number. In contrast, 0.5 leads to a 50 % step reduction in the number of elements. Each Gaussian starts with $n_g(0) = 100$ points. The number of points for Gaussian g at step t is $n_g(t) = \lfloor n(0)r_g^t \rceil$, but the standard deviation is kept stable with $\sigma = 0.4$. A temporal dataset is generated for 4 successive update steps.

3.4.2.2 Citation Graphs

A citation graph $G = (V, E)$ is a directed acyclic graph (DAG). V represents the a set of documents, like scientific papers, patents, law articles, blog posts, which are supposedly immutable. E is the set of directed edges, with $e = (a, b) \in E$ meaning that the document a is referring to document b, implicitly but assuming that a is newer than b.

Graphs are data structure that are difficult to represent in 2D, because they of their sparsity and the distribution in power-law of their node degree, which leads to a small number of strongly connected nodes, and a large number of weakly connected nodes. Nonetheless, citation graphs, as well as other real-world graphs, organise into local communities which are interesting to study.

We propose to study the evolution of research communities over time by embedding the documents published each year. The embedding obtained for year t is reused for year $t + 1$, which would be used in turn to build the following embedding. The DBLP dataset version 12 [27] was used for this purpose. This dataset corresponds to the citation graph of scientific papers around the topic of computer science. It contains 4.894.081 papers and 45.564.149 citing relationships. Metadata are available for the majority of the documents, providing information such as title, publication date, abstract, authors information, conference or journal reference, reference links, and keywords. Keywords also called *field of study*, are automatically extracted according to the method described in [44].

3.4.2.2.1 Graph preprocessing A preprocessing removes all existing cycles, as some papers are updated after the official publication date, adding a few citations. This phenomenom concerns a minority of papers, but creates undesirable loops. The cycles are removed using a DFS approach, removing any edges accessed twice by a DFS branch. The main connected component is then kept, removing all papers with no bibliography or belonging to an isolated community.

3.4.2.2.2 Node Sampling The citation graph considered is too large to be processed at once. We took advantage of the keywords to select a subset of papers related to a particular topic. We selected all documents related to *cryptography* and related topics, which corresponded to around 100.000 documents published between 1953 and 2020. The documents with less than one reference and less than one citation were removed, which left 70.000 documents published between 1953 and 2020, with the majority published between 2005 and later.

3.4.2.2.3 Extracting Distance Matrix from a Citation Graph A graph cannot be converted to tabular data used by t-SNE. Nonetheless, t-SNE and it-SNE use the distance between items and do not focus on particular features. Even if it is not possible to measure the euclidian distance between nodes on a graph, a distance matrix can be obtained.

The distance between nodes is not a great distance measure. The possible integer values are coarse measures, and the distance is not correctly defined for all pairs in a DAG. Plus, a document may quote unrelated documents from another discipline for illustrating its argumentation with other scientific views. The node could be at distance 2 of many papers on a completely unrelated field, just because of a single example.

A way to measure the document similarity is through bibliographic coupling [45]. Two documents sharing some of their references are coupled, even if there is no direct path from one to the other in the DAG. The strength of the coupling depends on the overlap size. As the graph is sparse, we extend the bibliographic coupling to undirect reference until distance 3. This strategy reduces the sparsity and improves the sensibility of the coupling.

The coupling strength is measured using the Jaccard similarity measure. For two documents, a and b, with respective reference sets A and B, the similarity between these two documents is defined as:

$$Sim(a,b) = \frac{|A \cap B|}{|A \cup B|} \tag{3.14}$$

A similarity is a value $s \in [0,1]$, while a distance is a value $d \in \mathbb{R}^+$. A distance close to 0 is equivalent to a similarity of 1, while a large disance to a similarity close to 0. Because the similarity behavior is the opposite of the one of a distance, a form of distance can be obtained by transforming the similarity:

$$D(a,b) = \frac{1}{Sim(a,b) + \xi} - \frac{1}{1+\xi} \tag{3.15}$$

where $\xi = 10^{-5}$ is a small constant, which avoids division by zero and limit the maximal distance to 10^5. Using equations (3.14) and (3.15), the citation graph can be appropriately transformed for the embedding algorithms.

3.4.3 Metrics

In this subsection, we define some of the metrics which would be used to evaluate the different parameter configurations.

3.4.3.1 Cost

The configuration of the embedding is evaluated in terms of cost, which is the quantity minimized by t-SNE:

$$\sum_{i \neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} \tag{3.16}$$

where P and Q correspond to the *intra*-probability matrices without considering the interaction with the previous embedding.

By comparing the cost of the embedding obtained with it-SNE to the reference one obtained with unmodified t-SNE, the comparison enables to study whereas the addition of a second cost term affect the optimization process.

3.4.3.2 Distortion

For an experiment where the items in $X^{(0)} \approx X^{(1)}$, a way to measure how well it-SNE places the point is to measure the distance between the inital and new position $Y^{(0)}$ and $Y^{(1)}$ and new position $Y^{(1)}$.

$$Err(Y^{(0)}, Y^{(1)}) = \sum_{i} |\mathbf{y}_i^{(0)} - \mathbf{y}_i^{(1)}| \tag{3.17}$$

3.5 Experimental Results

3.5.1 Evolution of Gaussians

Shrinking Gaussians This introductive experiment reproduces the protocol proposed in [42], were points are progressively attracted toward their Gaussian center of reference. The initial embedding $Y^{(0)}$ corresponding to $X^{(0)}$ is obtained with t-SNE, while all following embedding $Y^{(t)}$ for $t \geq 1$ are obtained with it-SNE using as a support the pair $(X^{(t-1)}, Y^{(t-1)})$.

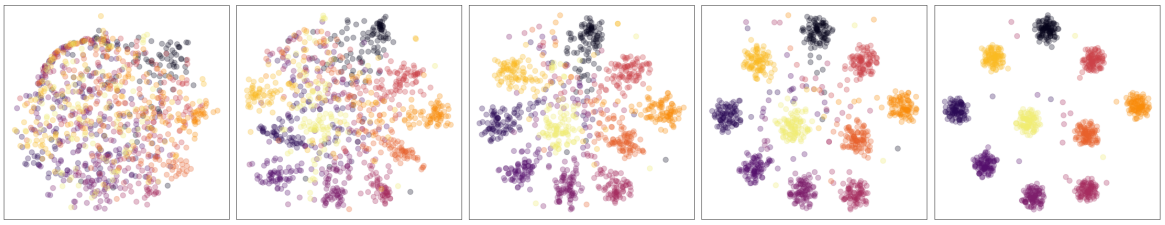


Figure 3.1: Shrinking gaussians. From left to right, steps 1, 3, 5, 7 and 9 are represented. Points are colored according to their gaussian center of reference. $\epsilon = 1.0$

Fig. 3.1 shows the result with it-SNE, which transforms undistinguishable groups into well-defined groups. Our approach works as well as dt-SNE, but while dt-SNE uses the $Y^{(t-1)}$ position to initialize $Y^{(t)}$, it-SNE restarts from random vectors. This difference frees our model from restrictions on the size and content of the dataset. This experiment was performed on freshly generated points sampled at each time step, leading to the same results as those presented in Fig. 3.1.

Growing Gaussians By reversing time direction, we get another set of experiments. The dataset starts with 10 Gaussians with $\sigma(0) = 0.4$, progressively increased to $\sigma(9) \approx 1.0$

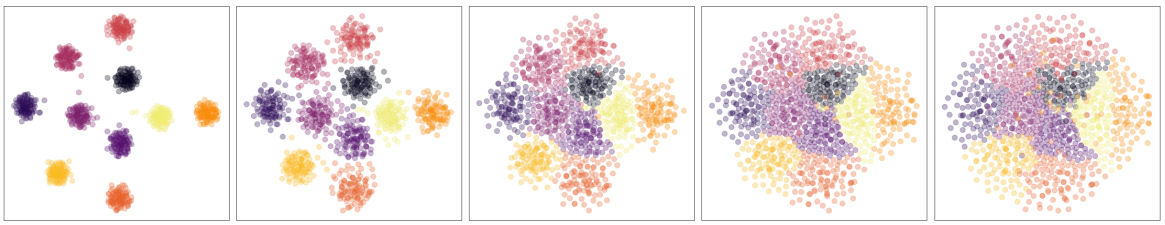


Figure 3.2: Growing Gaussians. From left to right, steps 1, 3, 5, 7 and 9 are represented. Points are colored according to their Gaussian center of reference. The penalty factor is set to $\epsilon = 1.0$

The results are represented on Fig. 3.2. This task is easier than the previous, as the first embedding starts with well-separated clusters. The next embedding support is of higher quality than in the previous experiment where the variance was larger. Even after moving to a noisier dataset (the last plot of *growing gaussians* has almost the same variance as the first plot of *shrinking gaussians*), the separation between items of different clusters is preserved even if clusters are not spaced from each other. A support embedding of good quality helps to guide items belonging to a noisy dataset, building a better embedding.

Change in Density The last visual result to present with gaussians focuses on density changes with a fixed σ. The number of samples per Gaussian changes at each step. For gaussian g at step t, the number of items generated around $\boldsymbol{\mu}_g$ is calculated as $n_g(t) = \lfloor n_0 r_g^t \rceil$.

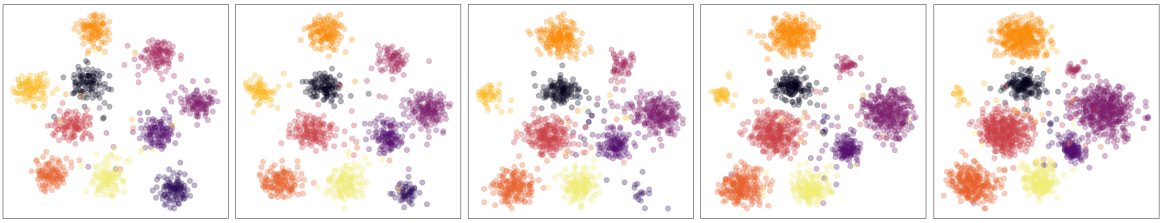


Figure 3.3: Evolving gaussians. 10 gaussians of various size with $\sigma = 0.4$. From left to right, time steps 0 to 4 are displayed. The penalty parameter is set to $\epsilon = 1.0$. Points are colored according to their Gaussian center of reference.

Fig. 3.3 illustrates the result of this process. At the start, all clusters have the same size and density and are spaced equally from each other. As time passes, some of the Gaussians grow in size, while others shrink. The area used by shrinking Gaussians decreases while growing Gaussians expand over the space available. The cluster positions are preserved despite the change of density unless too few items are present to allow the cluster aggregation.

3.5.2 Influence of ϵ

In this subsection, we discuss the impact of ϵ on the applied forces. For simplicity of the notation, the elements $p_{i,j}$ and $q_{i,j}$ correspond to $p_{i,j}^{(0,1)}$ and $q_{i,j}^{(0,1)}$, with i an element of the support dataset (0) while j an element of the dataset to embed (1). The same simplication is applied to $y_i^{(0)}$ and $y_j^{(1)}$.

3.5.2.1 Forces

The factor ϵ has an impact on Q and the gradient. The strength of the gradient is reduced, as ϵ plays in $\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\| + \epsilon^2\right)^{-1}$. As ϵ grows, the forces coming from the support embedding vanish.

The ϵ factor has an impact on the influence of items by changing the numerator and denominator of Q. When ϵ increases, the numerator $\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\| + \epsilon^2\right)^{-1}$ decreases for all item pairs. This value decays faster for items i and j that are close to each other than those which are not. The denominator of Q in (3.12), $V' = \sum_{i,j} \left(1 + \|\mathbf{y}_i - \mathbf{y}_j\| + \epsilon^2\right)^{-1}$ decreases when ϵ increases. As both numerator and denominator of Q decrease, the effective variation depends on the item proximity. Q increases for a pair of distant items and decreases for items that are close. The growth of ϵ reduces the variance, converging to $\lim_{\epsilon \to \infty} \mathbb{V}(Q) = 0$, and homogenize the value of $q_{i,j}$ to $\lim_{\epsilon \to \infty} q_{i,j} = \frac{1}{n_0 n_1}$.

Fig. 3.4 illustrates the evolution of Q values with ϵ, using the dataset with 10 gaussians. For the low value of ϵ, only items belonging to the same gaussian interact together (ie, 10% of the points). As ϵ grows, the weights of the nearest neighbors decrease to the profit of more

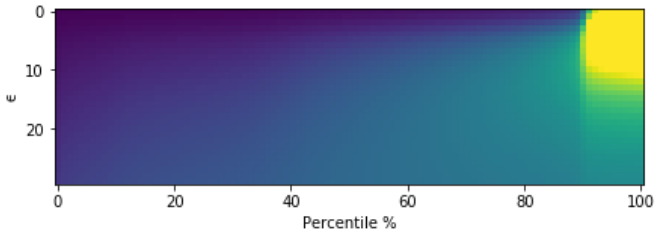


Figure 3.4: Percentile values of the $q_{i,j}$ for different values of ϵ. Colors are linear with the value of Q: dark colors correspond to value close to 0 while bright color to high value. The color saturates in yellow at $3n^{-2}$.

distant neighbors.

Because of the kernel asymmetry between P and Q, the reduction of variance and the convergence to the mean of Q leads to two different behaviors, depending on the closeness of two items. The items are divided into two classes based on the value of $p_{j,i}$. The closer neighbors with $p_{j,i} > \frac{1}{n_0 n_1}$ are considered as the nearest neighbors while the other distant neighbors. For nearest neighbors, the artificial distancing created by ϵ lowers the output probability $q > q^\epsilon$, for $q^\epsilon = q(\epsilon > 0)$ and $q = q(\epsilon = 0)$. P and Q's difference is $p - q^\epsilon > p - q$ is then larger, which leads to larger attractive forces from the close neighborhood. The opposite effect happens for distant neighbors where $q < q^\epsilon$, which leads to $p - q > p - q^\epsilon$, generating repulsive forces.

To summarize, on the one hand, the forces are globally lowered as ϵ impacts the gradient, and on the other hand, the discrimination between nearest and distant neighbors grows as ϵ amplifies the asymmetry.

3.5.2.2 Displacement from Origin

A way to study the two contributions can be done by looking at the ability of *it*-SNE to recover the exact embedding positions.

For a dataset $X^{(0)}$, first is computed $Y^{(0)}$ with:

$$Y^{(0)} \leftarrow t\text{-SNE}(X^{(0)})$$

followed by:

$$Y^{(1)} \leftarrow it\text{-SNE}(X^{(0)}, X^{(0)}, Y^{(0)})$$

.

Fig. 3.5 shows the cluster conformation for two different ϵ. The clusters are correctly matched, both in classes and in positions for $\epsilon = 1$, as $Y^{(1)}$ masks $Y^{(0)}$. However, for $\epsilon = 25$, while the cluster classes are corectly matched, they are distant from the original position.

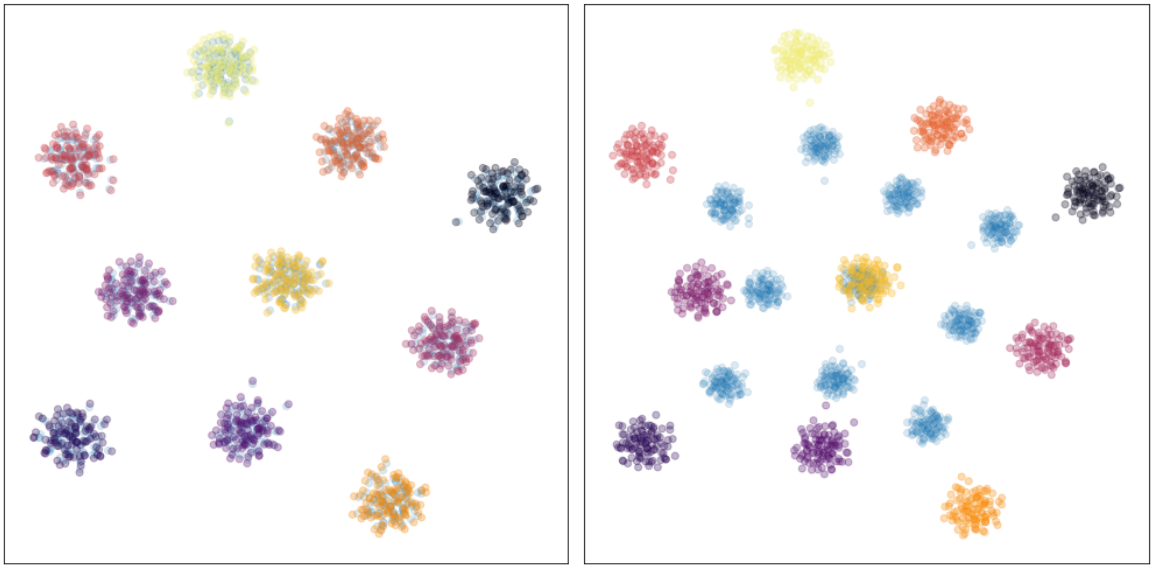


Figure 3.5: Mapping of gaussians with standard deviation $\sigma = 0.4$. The embedding $Y^{(0)}$ is colored in light blue on each plot, while items of $Y^{(1)}$ are colored according to their cluster of reference. Left: $\epsilon = 1$, right: $\epsilon = 25$.

To study the transition between the two conformations, we look at the distortion $Err(Y^{(0)}, Y^{(1)})$, which measures the distance between the initial and final position.

Fig. 3.6 shows the impact of an increase of ϵ over the item locations. For low ϵ, the error is almost 0. The average distance error for $\epsilon = 0$ is 0.204, while the average distance to the first nearest neighbor is 0.218, for an embedding of diameter 32.1. With increasing values of ϵ, the distances between initial and final positions grow. The right part of Fig. 3.5 illustrates the situation. The repulsive forces between distant neighbors are stronger than the attraction of the nearest neighbors. This pushes the clusters even furthen away from each other, increasing the distance from the initial position. The maximal distortion is reached around $\epsilon = 25$, a value of the order of the embedding diameter.

After this maxima, the distortion error decreases to stabilize at a value of 2.6. Compared to the diameter of one cluster presented in 3.5, the value is relatively similar, around 2.88. The clusters overlap, but the forces are not strong enough to accurately bring them to their exact location.

3.5.2.3 Cost

it-SNE tries to minimize two costs at the same time. They might not be compatible, with opposite gradient directions. The intra cost allows assessing the embedding quality to see if the embedding could reach a correct minima, or if the inter forces constrained the embedding to a non-optimal state.

Fig. 3.7 presents the results for the same conformations as in Fig. 3.6. The cost is slightly higher to start with, but it decreases as ϵ increases to arrive at a local cost minima. This local minima corresponds to the distortion maxima of Fig. 3.6. The conformation $\epsilon = 25$ is a

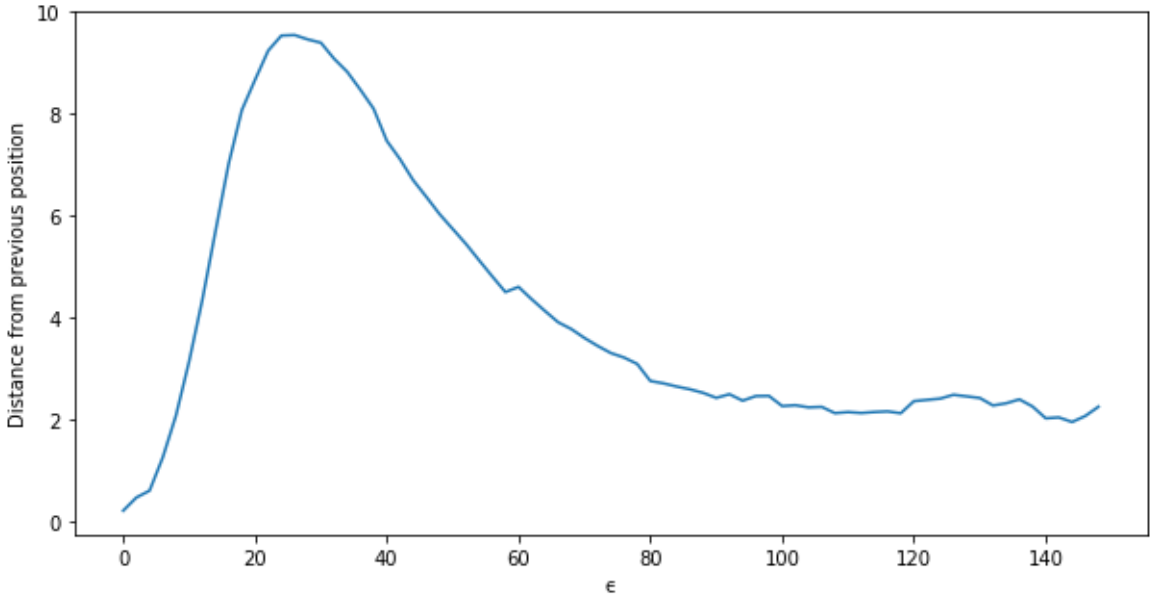


Figure 3.6: Evolution of the embedding error $Err(Y^{(0)}, Y^{(1)})$ with ϵ using 10 gaussians with standard deviation $\sigma = 0.4$.

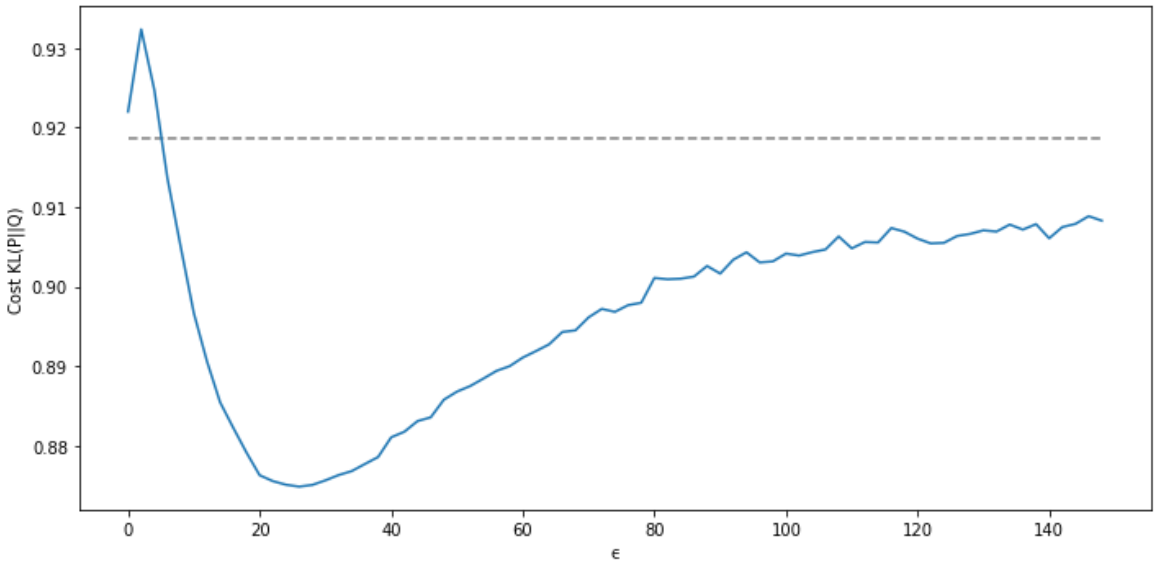


Figure 3.7: Evolution of Kullback Leibler cost with increasing ϵ. Gaussians with standard deviation $\sigma = 0.4$. The dashed line corresponds to the cost of embedding $Y^{(0)}$ obtained with the regular t-SNE

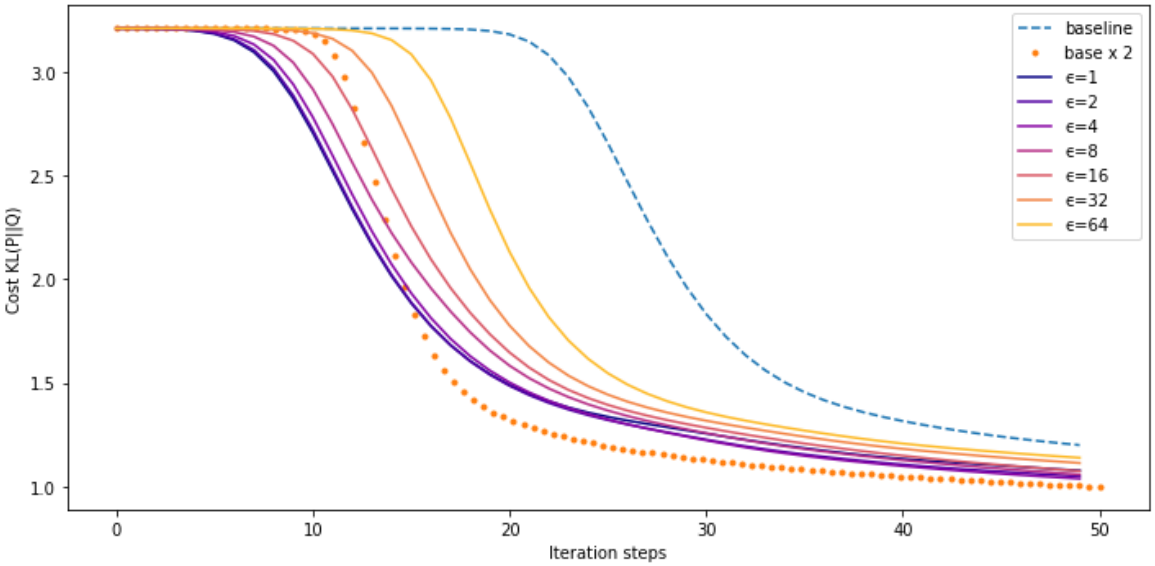


Figure 3.8: Evolution of the Kullback Leibler cost over training time, with 10 gaussians of standard deviation $\sigma = 0.4$. The yellow to dark lines correspond to it-SNE for several value of ϵ The blue dashed line corresponds to the cost evolution for t-SNE. The dotted line corresponds to baseline with t-SNE, speeded by a factor 2.

more stable configuration than the initial one. For larger values, the forces' strength decrease, but stay below the baseline cost, corresponding to the support configuration. It is to note that the cost difference between the lowest and highest cost value in Fig. 3.7 is relatively small. All conformations are relatively good, some a little bit more than others.

3.5.2.4 Convergence Speed

In our protocol, the number of training step was fixed to control the computational time. For a support embedding of n_0 items and a new dataset to embed of n_1 items, the t-SNE cost is proportional to n_1^2, while the cost for it-SNE is proportional to $n_1^2 + n_0 n_1$. In our experiments $n_0 = n_1 = 1000$ which means that the number of operations performed by it-SNE is twice the number of t-SNE.

Fig. 3.8 shows the cost evolution for several values of ϵ. All curves start with a plateau, which corresponds to when the learning rate is not boosted enough to lead to significative changes per steps. The lower ϵ is, the shorter the time spent on the plateau is. A decay part follows the plateau, which smoothly slowed down until convergence.

The dotted orange line allows comparing t-SNE with it-SNE on the number of computational operations. t-SNE is faster than it-SNE, but the difference between the two is not very large.

3.5.3 Citation Graph Embedding

Gaussian clusters are easy to study as the parameters are fully controlled. Citation graphs are selected to illustrate a real-world example of it-SNE capabilities.

Because a scientific article refers to relevant papers in its field, this type of dataset presents

local community structures. The evolution of the number of researchers is leading to an expansion of knowledge in many fields, and the creation of new ones. Other fields tend to disappear, because of a lack of support from the scientific community or an absence of new discoveries. The study of these phenomenom allows to retrace history and reconstruct the science phylogeny [46].

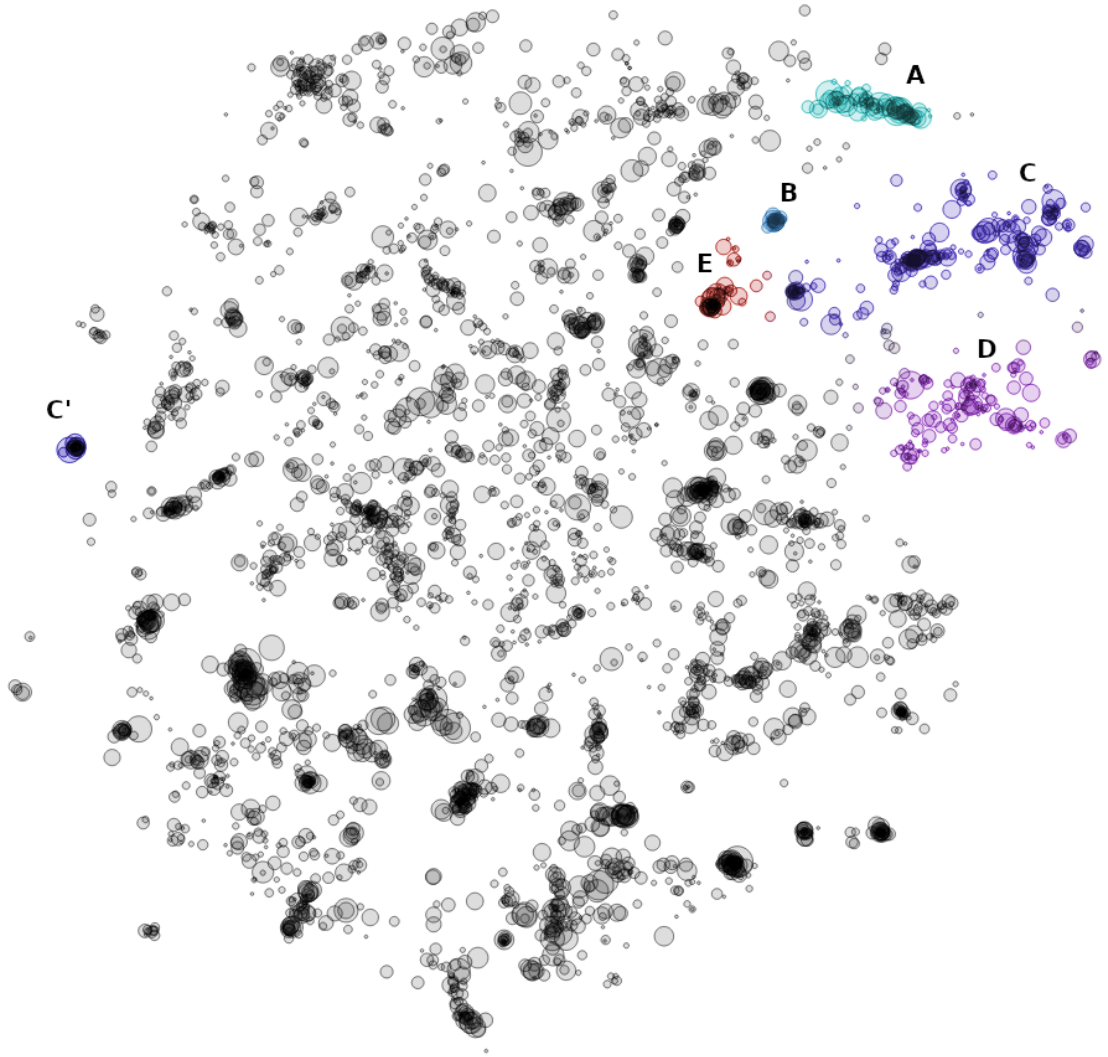


Figure 3.9: Citation graph of cryptographic papers in 2010, with some highlighted clusters. A: Hashing, B: Network Code, C and C': Biometry, D: Watermarking/Data Hiding, E: Passwords. The item size is proportional to the number of citations.

Fig. 3.9 represents the papers published in the cryptography/security field in 2010. Papers groups together to form connected clusters with various shapes, sizes and densities. To the default of a clustering algorithm, some groups have been highlighted and labelled by hand with the help of documents' title and keywords.

Cluster A, with *Hashing*'s general topic, is compact with items well connected to each others. Cluster D about *Watermark* is more diffuse than A, but items are still grouped together.

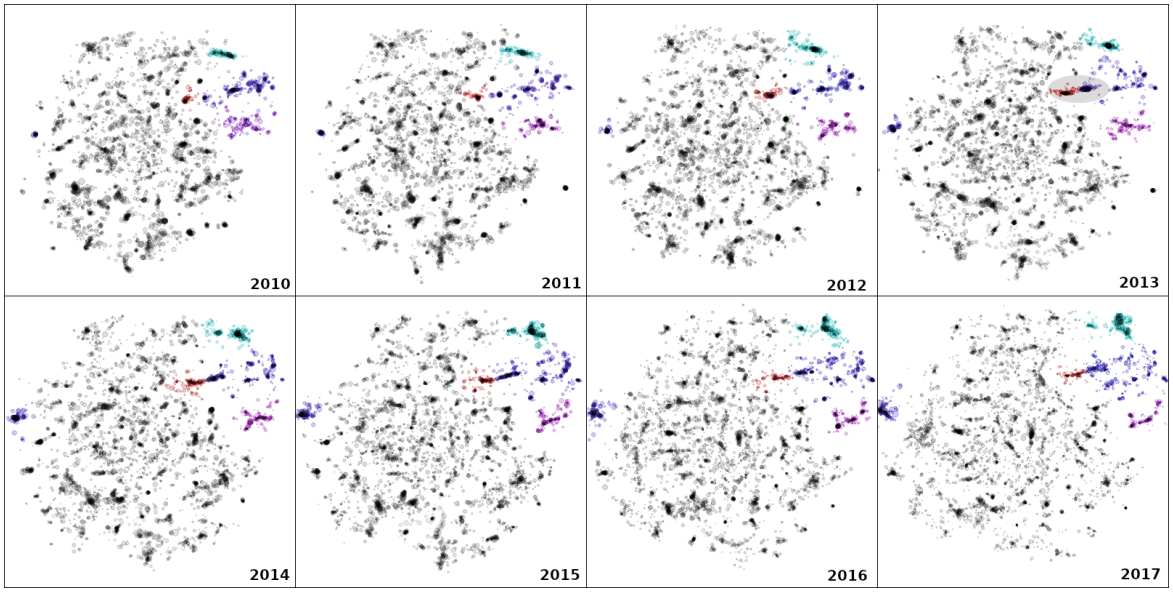


Figure 3.10: Citation graph of cryptographic papers, published between 2010 to 2017. Clusters are highlighted according to the previous figure coloring scheme. Size of points is proportional to the logarithmic number of citations. Dark areas correspond to highly grouped papers. The fusion between Biometry and Password clusters in 2013 is shaded in grey.

The *Biometry* cluster C is composed of several sub-units of various density. Cluster B and E about *Network Code* and *Password* respectively are much more compact than the other presented.

The papers in each cluster are mostly in phase with the general cluster topic, showing local unity. Nonetheless, the reverse is not valid: a cluster about a particular topic does not englobe all documents related to it. A keywords may correspond to two different ideas, or two distinct communities may work on different aspects of the problem. For instance, this is the *Biometry* field case, which occurs twice in various embedding locations. There is one large cluster on the right and a smaller one on the left (denoted C and C' respectively). While the large cluster C is about general biometric recognition methods, the C' is focussed on authentication scheme, mixing *Biometry*, *Password* and *Authentication* topics together.

Scientific communities are dynamic and adapt to new trends. Fields emerge, grow, interact, split, and disappear. *it*-SNE allows reusing these initial cluster positions for the next subsequent embedding, which allows tracking such dynamics.

Fig. 3.10 presents the evolution of the embedding 3.9 from 2010 to 2017, with the same color highlighting. The general shape of the embedding is stable over time, with clusters' positions preserved. A small drift of the clusters occurs, which is noticeable after a few embedding steps. The growth and shrink of some clusters is visible, such as for the second biometry cluster on the left which expands over the years. Merges are also visible, such as for the *Biometry* and *Password* clusters, which first merge in 2013.

To have a better view of the evolution, Fig. 3.11 shows an enlarged view of the top right

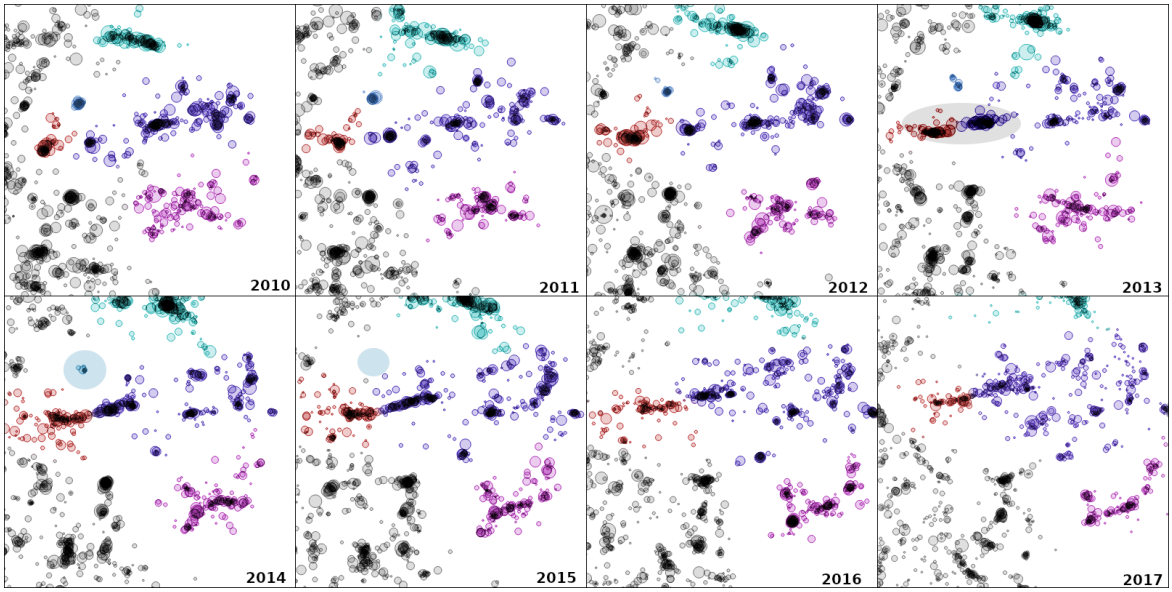


Figure 3.11: Citation graph of cryptographic papers, published between 2010 to 2017. Clusters are highlighted according to the previous figure coloring scheme. The size of points is proportional to the logarithmic number of citations. Dark areas correspond to highly grouped papers. The fusion between Biometry and Password clusters in 2013 is shaded in grey.

area of the embedding, where *Biometry* and *Password* field of study clusters are. In this area, different types of phenomena occur. There are stable clusters present, such as the *Hashing*, *Biometry*, and *Watermark*, with constant size and density. *Password* cluster growths in size, while *Network Coding* disappeared in 2015. The *Biometry* and *Password* groups have been interacting with each other and began merging in 2017.

As the clusters have been human extracted, no metric measure has been tested. Tables listing the most cited paper for each year for the different clusters are presented in the appendix (Tables 3.1, 3.2, 3.3, 3.4, 3.5, 3.6). The articles presented in these tables are very consistent with the theme of the cluster from which they originate.

3.6 Discussion

3.6.1 Complexity

The normal complexity of t-SNE is in $\mathcal{O}(n^2)$, where n is the number of items. While some optimization exists when the dataset is tabular, like the Barnes-Hut optimization [47] which reduces the cost to $\mathcal{O}(n \log n)$, for data like graphs from which a distance matrix can be obtained, this quadratic cost is prohibitive.

If the dataset is cut into k equivalent pieces of $m = \frac{n}{k}$ pieces, the algorithm would run in $\mathcal{O}(2km^2) = \mathcal{O}(\frac{2}{k}n^2)$, where the $2\times$ stands for the two gradient parts. This reduces by a factor $\frac{k}{2}$ the complexity. The complexity here measures the average number of operations for one run of an iteration step. However, as the algorithm uses gradient descent, a convergence criterion governs the total number of steps. Intuitively, the number of steps required to converge for a large dataset seems larger than for a smaller dataset. The decomposition of the dataset into pieces would reduce the effective computational time.

Concerning the memory requirements, the normal t-SNE requires a storage space of $2n^2$, necessary for the matrix P and Q. Using it-SNE with a dataset split into k pieces, 4 matrix of size m^2, $(P^{(t)}, Q^{(t)}, P^{(t-1,t)}, Q^{(t-1,t)})$ are needed to compute the embedding. The total amount of memory required is then $4m^2 = 4\left(\frac{n}{k}\right)^2$, which is more interesting, as $\frac{k^2}{2}$ reduces its cost. The computational time can be extended on a machine, but not its memory. Our proposed method can be helpful as way to map large datasets by cutting them into smaller pieces.

3.6.2 Selecting ϵ

Speedup The parameter ϵ controls the applied forces and the gradient strength. A low value of ϵ creates strong forces which leads to fast convergence. Nonetheless, forces prevent items to move to other locations. An increase of ϵ would relax the system and help an item to arrive on a low energy state. As for the *early exaggeration* trick, it would be beneficial to start with a small value of ϵ and then finish with a larger value.

ϵ and Perplexity The perplexity governs the number of neighbors taken into account. The numerator of P in equations (3.1) and (3.10) grows with a perplexity increase for all items. The denominator grows too, and leads to a decrease of P for the nearest neighbors. It repercutes on Q which needs to decrease. The distances between Y's increase with larger perplexity.

If the support embedding has a different perplexity than the target perplexity, a large ϵ may help to adapt to the new perplexity, by relaxing forces strength. The clusters would be attracted to nearby position, and the intra-forces would arrange the local shape.

3.6.3 Adaptation to Large Changes of Density

Our experiments have been done with temporal datasets with constant or slowly evolving size. The use of a support dataset of highly different size may constrain the system optimization. For a fixed perplexity, the diameter of an embedding grows with the size of the dataset in $(n)^{\frac{1}{d_e}}$. Two datasets of different size would have a radius of $r^{(0)}$ and $r^{(1)}$. The items located in the middle of the embedding can be correctly matched with the support items. However, for peripheral items, there would be a gap of $|r_0 - r_1|$. *it*-SNE would create a distortion, constraining items of (1) to expand if $r_0 > r_1$, which would increase the interdistances between items in $Y^{(1)}$. For $r_0 < r_1$, a shrinking would occur leading to the same distortion. As intra forces of (0) do not play any role, a trick to free from this constraint would be to scale $Y^{(0)}$ into $Y^{(0*)}$, using the scaling ratio $s = \left(\frac{n_1}{n_0}\right)^{\frac{1}{d_e}}$, such as $Y^{(0*)} = sY^{(0)}$. This would help for datasets of similar densities. For different densities, local distortions would still occur.

3.6.4 Using more than one Support Embedding

The method proposed to use one embedding to support the generation of a new one. A natural question arises about the possiblity of using the support of two or more embeddings. This case may happen if two embeddings for $t_0 < t_2$ have been obtained but not for t_1 yet, with $t_0 < t_1 < t_2$. Intuitively, the constrains engendered by the two embeddings might be equivalent to a single one. If more embeddings were to be taken into account, all grouped embedding cost must not prevent the intra forces from playing their role.

For a dataset $X^{(k)}$ taking support of datasets $\mathcal{X} = \{X^{(i)}\}_{i=1:k-1}$ with respective embeddings $\mathcal{Y} = \{Y^{(i)}\}_{i=1:k-1}$, the cost could be rewrited as:

$$C_{tot}^{(k)} = C^{(k)} + \frac{1}{k-1}\sum_{i=1}^{k-1} C^{(i,k)} \tag{3.18}$$

This adaptation allows generating embedding in between two existing embeddings. Another use of this adaptation is the multivariate case, like for geolocation coordinates, where the new embedding might take the support using multiple non-equivalent datasets. For $\mathcal{X} = \{X^{(i)}\}_{i=1:k-1}$ with relative importance $W = \{w_i | w_i > 0\}_{i=1:k-1}$, the weighted cost would have the form:

$$C_{tot}^{(k)} = C^{(k)} + \frac{1}{\sum_{i=1}^{k-1} w_i}\sum_{i=1}^{k-1} w_i C^{(i,k)} \tag{3.19}$$

Note that the use of multiple support embeddings increases the cost linearly with the total number of items. Nonetheless, if the nearest datasets are too small to serve as a support, the use of more than one embedding may helps to preserve embedding knowledge and enhance long term coherency.

3.6.5 Binary Computation

To create several coherent embeddings for a succession of datasets, the process can be speeded-up by distributing the embedding tasks. If there are k datasets to embed, instead of computing the embeddings in a sequential way, using the support of $t-1$ to compute t, the use of another more distant support would help. The closer the support, the better it would be, as the distribution difference between two neighbor datasets is expected to be lower than for distant datasets.

The computation starts with an initial embedding for $\lfloor \frac{k}{2} \rceil$. Then, the left and right intervals are divided in their middle. An embedding is issued for $\lfloor \frac{k}{4} \rceil$ and $\lfloor \frac{3k}{4} \rceil$. Then, the embedding for subset $\lfloor \frac{1}{8}k \rceil$ can be computed using the support of $\lfloor \frac{k}{4} \rceil$, and $\lfloor \frac{7k}{8} \rceil$ using the support of $\lfloor \frac{3k}{4} \rceil$. For the middle parts $\lfloor \frac{3}{8}k \rceil$ and $\lfloor \frac{5}{8}k \rceil$, their respective embedding is computed using two support embeddings, respectively using $(\lfloor \frac{k}{4} \rceil, \lfloor \frac{k}{2} \rceil)$ and $(\lfloor \frac{k}{2} \rceil, \lfloor \frac{3k}{4} \rceil)$. The procedure is repeated recursively until all embeddings have been obtained.

This decomposition allows to speedup the process from $\mathcal{O}(k)$ to $\mathcal{O}(\log_2(k))$.

3.7 Conclusion

This paper presents a method adapting t-SNE algorithm to reuse a previous embedding to generate a new one. Compared to the base method t-SNE, an additional cost term is added. This cost links the new items to embed to the support embedding, creating attractive forces. These forces enable the similar items from the support and current datasets to be located on the same embedding area. Clusters are coherent in the location from one embedding to the other, enabling the reuse of a classification algorithm on both.

it-SNE was tested on two datasets. The first used synthetic gaussians forming dense clusters, evolving in density and size over time. The second was the scientific citation graph restricted to cryptography related papers, with small, sparse communities. The algorithm was successful at preserving the cluster locations in both experiments, while preserving t-SNE embedding aspect.

Compared to t-SNE, the computational complexity and the memory requirement of *it*-SNE are doubled. Nonetheless, the use of a support embedding speedups the convergence process of *it*-SNE. The total number of operations of *it*-SNE is, in practice, equivalent to t-SNE.

We proposed two extensions: the first to the multivariate case and the second to distribute the embedding computation. One unsolved problem yet is the adaptation to highly different densities, as t-SNE mechanism tries to keep average distance between neighbors constant, which leads to an expansion of the embedding with increasing dataset size.

it-SNE can be used for many purposes, such as monitoring, anomaly detection, network analysis, allowing to track the evolution of clusters in a low dimensional space. The method is not restricted to temporal datasets and could be used to study the impact one variable's

impact on the dataset distribution.

Appendix: Most cited papers

The following tables list the most cited document per year per topic of the different clusters highlighted in Fig. 3.9. Some titles have been truncated.

Table 3.1: Hashing

Year	Title
2010	Semi-supervised Hashing for Scalable Image Retrieval
2011	Minimal Loss Hashing for Compact Binary Codes
2012	Image Signature: Highlighting Sparse Salient Regions
2013	Inter-media Hashing for Large Scale Retrieval
2014	Supervised Hashing for Image Retrieval
2015	Supervised Discrete Hashing
2016	Deep Supervised Hashing for Fast Image Retrieval
2017	Learning Discriminative Binary Codes

Table 3.2: Network Coding

Year	Title
2010	Secure network coding over the integers
2011	Secure Network Coding on a Wiretap Network
2012	Cooperative Defence Against Pollution Attacks
2013	An Efficient Homomorphic MAC with Small key Size
2014	A Lightweight Encryption Scheme for Network-Coded Mobile

Table 3.3: Biometry

Year	Title
2010	Unobtrusive User-Auth on Mobile Phone
2011	A survey on Biometric Cryptosystems and cancelable biometrics
2012	Touch me once and I Know it's you
2013	Touchalytics: On the Applicability of Touchscreen Input
2014	Image quality Assessment for Fake Biometric Detection
2015	Deep Representation for Iris, Face and Fingerprint
2016	Continuous User Authentication on Mobile Devices
2017	MagNet: A Two-Pronged Defense against Adversarial Examples

Table 3.4: Biometry and Authentication Schemes

Year	Title
2010	An Efficient Biometrics-based Remote User Authentication Scheme
2011	Cryptanalysis and Improvement of a Biometrics-based Remote
2012	A secure Authentication Scheme for Telecare Medecine
2013	A Temporal-Credential-Based Mutual Authentication
2014	A Novel User Authentication and Key Agreement Scheme
2015	Robust Biometrics-Based Authentication Scheme
2016	An efficient User Authentication and Key Agreement Scheme
2017	Anonymous Authentication for Wireless Body Area Networks

Table 3.5: Watermark

Year	Title
2010	Review: Digital Image Steganography
2011	Reversible Data Hiding in Encrypted Image
2012	Separable Reversible Data Hiding in Encrypted Image
2013	Digital Image Forgery Detection using Passive Techniques
2014	Reversibility improved data Hiding in Encrypted images
2015	RAISE: a Raw Images Dataset for Digital Image Forensics
2016	Reversible Data Hiding: Advances in the Past Two Decades
2017	Fragile Image Watermarking with Pixel-wise Recovery

Table 3.6: Password

Year	Title
2010	Encountering Stronger Password Requirements
2011	Of Passwords and People: Measuring the Effect of Passwords
2012	The Quest to Replace Passwords
2013	Patterns in the Wild: a Field Study of the Usability of Pattern
2014	It's a Hard Lock Life: A Field Study of Smartphone Unlock
2015	"... No one Can Hack My Mind": Comparing Expert
2016	Who are you? A Statistical Approach to Measuring User Auth.
2017	Zipf's Law in Passwords

Part II

Clustering Sparse Bipartite Graphs

CHAPTER 4

Tagged Documents Co-Clustering

Tags are short sequences associated with a resource, such as as music, image, book, video, patent, shoping item, and many others. For non-textual resources, they can be used to describe the content of a document. These tags are useful for machine information retrieval systems to access quickly a document or use recommender systems to suggest similar items. This short description allows it to guess if a resource would be relevant for a user. However, due to the limited number of tags per document, the match between a user query and the document's tags is often low. In this paper, we propose a methodology to cluster tags into conceptual groups, which could be used to refine a query. As keywords follow Zipf's law distribution, data are preprocessed to remove power-law effects and enhance the context of low-frequency words. Then, a hierarchical agglomerative co-clustering algorithm is proposed to group together the most related tags into clusters. The capabilities were evaluated on a sparse synthetic dataset and a real-world tag collection associated with scientific papers. The task being unsupervised, we propose some ending criterion for an optimal partitioning.

4.1 Introduction

Tags are words or short sequences associated with a resource or a document. Depending on the context, the role of a tag differs. They could be used to describe feeling, category, source, content, ownership and others [48].

The tags associated with a document form a bag of words, where the order does not matter. Tags can be weighted or ranked by relevance, helping the machine or user to know which are the most accurate or specific to the document.

Tags can be obtained with two different approaches: they can be machine extracted, where an algorithm process the resources and extract some of their characteristics [27,49]; or they can be handcrafted by *experts* or *non-experts*. In the expert case, the vocabulary is controlled, and the results would be comparable to machine extracted tags. In contrast, *folksonomy* corresponds to non-expert tags, leading to a large corpus with redundant or mispelt tags.

The corpus is often represented using the vector space model, where a document is represented by a sparse binary vector, where a 1 encode the presence of a tag within the document. The document's tags cannot be rank by importance because a document is described by a binary vector. Tags can only be ranked within the corpus, and often shows power-law distribution [50], making their analysis difficult because of the scarcity of frequent keywords and an abundance of unfrequent tags. Additionally, their number per document is relatively smaller than for usual textual documents, as the goal of tags is to provide a synthetic view of the document. This reduces the precision of an analysis when a tag is missing, as there is not enough tag redundancy to compensate for the absence of this tag.

A way to improve the tag vector representation is to rely on an external database [51] like WordNet [52] or Wikipedia [53] to add additional related tags to enrich the initial tag vector description. Another possibility is to rely on a probabilistic model [54], modeling keywords distribution based on available data.

Tags can be use for information retrieval and recommendation. For a machine, it allows proposing related documents using tag co-occurrences. For a human, tags can be used to create *tag clouds* [55] to help a user to refine its query by suggesting related keywords. A tag cloud displays the best co-occurring tags, adjusting the size, color, opacity of the tags to the context. Nonetheless, the tag cloud utility is limited due to the amount of irrelevant unorganized information [56,57]. Rather than ranking tags against an initial query, another option is to cluster tags [58], grouping them by context.

Many algorithms, trying to cluster keywords alone, without clustering documents [57,59,60] exist. These approaches do not take advantage of the duality between samples and features. For our particular setup where documents are succinctly described, it seems more relevant to use an algorithm clustering samples and features at the same time to improve the clustering quality.

Co-clustering approaches cluster both on rows and columns together. Many approaches focus on the bipartite graph representation [61–63] of keywords and documents. [63] proposed to use the spectral decomposition to cluster samples and features using their eigenvector representation. The work [64] proposed using information theory methods, which given an initial partitioning alternates between row clusters refinement and column clusters refinement.

There are two main problems in clustering: defining the target number of clusters, and defining the rules for cluster assignment. Concerning the cluster count, we propose a hierarchical agglomerative algorithm that stores the merging operation history. It free us from setting *a priori* a specific number of clusters. Nonetheless, we suggest a stopping criterion to select an optimal partitioning. We follow the probabilistic and information theory approaches to cluster tags with limited information available. We follow a co-clustering approach to take advantage of the synergies between documents and keywords clustering.

In this paper, we describe in the first part, a procedure to enhance keywords and documents context without the use of an external database. Next, the algorithm and its cost function are detailed. The details about datasets and metrics used are presented in the experimental setup section, followed by the experimental results. Then, the article ends with a discussion and a conclusion.

4.2 Enlarging Document Context

4.2.1 Documents Keywords Matrix

Be $X = \{x_i\}_{i=1:n}$ the set of n documents and $Y = \{y_j\}_{j=1:m}$ the set of m keywords, which occur in the set of documents. Using the vector space model, a document is represented under the form of a binary vector $\mathbf{x}_i = [x_{i,1}, ..., x_{i,m}]$ where a 1 encodes the keyword's presence while a 0 its absence in document i. The same representation applies to keywords, represented as $\mathbf{y}_j = [y_{j,1}, ..., y_{j,n}]$, where a 1 encodes the occurrence of keyword j in a document.

The collection of document vectors is often represented in a matrix form. The documents-keywords matrix is $M \in \{0,1\}^{n \times m}$, with $M_{i,j} = x_{i,j} = y_{j,i}$, where rows represent documents and columns the keywords.

4.2.2 Similarity Matrix

In practice, the documents-keywords matrix is very sparse as a document is assumed to be tagged with only a few relevant keywords. For a keyword, the sparsity leads to very few keyword co-occurrence pairs with very low weight.

In our particular case, we assume that keywords are exact and relevant, accurately extracted by the algorithms or experts, and none due to mistakes. This assumption simplifies the task of enhancing the keywords-documents matrix. Rather than searching for incorrect pairs first for data cleaning, all the keywords pairs are taken into account to improve the

keywords-document matrix.

To improve the matrix, we take advantage of the co-occurrence matrices, one measuring the document's similarity, the other measuring the keyword's similarity.

A cosine-like similarity is used to compute the similarity between pairs of documents and keywords, $S^{(X)}$ and $S^{(Y)}$ respectively, taking values in $[0, 1]$. Instead of running the cosine similarity on binary vector, the values are adjusted by the relative frequency. For document i, each keyword is weighted by its frequency c_k^{-1} where $c_k = \sum_i M_{i,k}$, to ensure that frequent keywords count less than infrequent ones. The same normalization is performed on keyword vectors, normalized by the number of keywords within each document $r_k = \sum_{j=1}^{m} M_{k,j}$. The document normalization has almost no effect, as the number of tags per documents is relatively homogeneous, but affect a lot keyword normalization, as they follow a power-law distribution.

The document similarity is defined as:

$$S_{i,j}^{(X)} = \frac{\sum_{k=1}^{m} \frac{M_{i,k} M_{j,k}}{c_k}}{\sqrt{\left(\sum_k^m \frac{M_{i,k}}{c_k}\right)\left(\sum_k^m \frac{M_{j,k}}{c_k}\right)}} \tag{4.1}$$

The equation (4.1) is adapted to compute $S^{(Y)}$ by making the sum over the rows rather than the columns (i.e., $\sum_{k=1}^{n} \frac{M_{k,i}}{r_k}$), and normalized by the number of keywords describing the document r_k. The normalization is equivalent to the inverse document frequency used in TF-IDF to score keywords by relevance, enabling to focus on singular keywords rather than frequent one.

4.2.3 Transition Matrix

The similarity matrix represents on a scale from 0 to 1 how well two elements are related. The closeness is characterized by a value close to 1 while unrelatedness by 0. Similar keywords can be either synonymous or occurring in the same context. In some way, the similarity matrix represents the co-occurrence laws.

To exploit these relationships, we suggest transforming the similarity matrices into the form of Markov transition probability matrices, where $T_{i,j} = Pr(j \to i)$ is the probability to move to state j starting from i. Using the product between T and M, we would obtain a vector with weights representing the probability of obtaining a given keyword given the initial tags.

A simple normalization of S is not enough to obtain T. The normalization of S ensures that the mass-distributed by an item to its neighbors is preserved after transformation, ie $|T\mathbf{u}^{\dagger}| = |u|$. However, it does not ensure that the mass is fairly attributed. Highly frequent keywords are more receptive than infrequent ones, as $\sum_j Pr(j \to i)$ is larger for frequent items.

To rebalance the mass attribution, the matrix T is obtained by bi-stochastization, leading to $T = T^{\dagger}$, ie $Pr(i \to j) = Pr(j \to i)$. We use the Sinkhorn-Knopp algorithm [65], which alternates between searching the best column normalization vector $\mathbf{c}$ and the best row normalization vector $\mathbf{r}$, repeating until convergence $\mathbf{c} = (S\mathbf{r})^{-1}$ and $\mathbf{r} = (S^{\dagger}\mathbf{c})^{-1}$. The transition matrix is obtained by:

$$T = \mathcal{D}(\mathbf{r})S\mathcal{D}(\mathbf{c}) \tag{4.2}$$

where $\mathcal{D}(\mathbf{z})$ is the diagonal matrix with element $D_{i,i} = z_i$.

4.2.4 Matrix Smoothing with Mass Preservation

The two transition matrices are used to smooth the initial documents-keywords matrix using:

$$M^* = T^{(X)}MT^{(Y)} \tag{4.3}$$

The detail of a term of M^* obtained following (4.3) is:

$$M^*_{i,j} = \sum_{k=1}^{n}\sum_{\ell=1}^{m} Pr(x_i|x_k)Pr(y_j|y_\ell)M_{k,\ell} <! -- \tag{4.4}$$

The term $M^*_{i,j}$ of (4.4) corresponds to all possible transitions to a document i and keywords j from all possible document-keyword pairs $M_{k\ell}$.

The application of $T^{(Y)}$ preserves the mass on the rows while $T^{(X)}$ preserves it on columns. After application of both, only the global mass is preserved $\sum_{i,j} M^*_{i,j} = \sum_{i,j} M_{i,j}$.

This transformation redistributes the weights for documents and keywords without changing the global mass of the system. The matrix M^* will be used for the co-clustering instead of the binary matrix M.

4.3 Clustering Maximizing Information

4.3.1 Agglomerative Clustering

An agglomerative clustering algorithm iteratively aggregates items from X into groups, leading to a hard partitioning. The algorithm starts with an initial partitioning where each item of X is alone in its own cluster, i.e. $\mathcal{C}^{(X)} = \{c_i = \{x_i\}\}_{x_i \in X}$. The clusters to merge are selected according to a cost function $D : C \times C \to \mathbb{R}^+$ which scores cluster pairs. The pair (c_i, c_j) with the lowest cost are merged together to form a new cluster $c_k = c_i \cup c_j$. The agglomeration process take $n - 1$ steps for the rows, where $n = |X|$.

The cost function affects the algorithm outcome [66,67]. The selection of a specific cost function depends on the assumption over the cluster shape. For instance, *single linkage* focusses on merging clusters with the smallest gap $C(c_i, c_j) = \sum \min_{x_k \in c_i, x_\ell \in c_j} d(x_k, x_\ell)$, without taking into account the cluster mass, while *complete linkage* focusses on merging clusters with the lowest maximal distance $C(c_i, c_j) = \sum \max_{x_k \in c_i, x_\ell \in c_j} d(x_k, x_\ell)$. It results in two different behaviors: *single linkage* is sensitive to noise, as it would create artificial bridges between clusters, while *complete linkage* is sensitive to outliers, preventing the merge of clusters containing some of them.

On the vector space model, the use of distance measures is not satisfactory [68], as the information per documents is too short to get an accurate representation. Instead of distance, the dissimilarity between clusters is measured using the divergence between their probability distribution.

A partitioning with one large cluster and many singleton clusters made of outliers is similar to a filtering algorithm. A clustering with such an outcome is not desirable as no *true* group exists. A partition must be composed of clusters with equivalent size, without high disparity. Some algorithms naturally take into account the cluster size. For example, in the case of *complete linkage*, where the larger a cluster becomes, the harder it is to merge as the maximal distance to other clusters tends to grow. When using a cluster probability distribution, all items within the clusters are represented by a single prototype independent of the cluster size.

To remediate to the fact that cluster prototypes do not include the knowledge of their size, we define our agglomerative algorithm cost as the product between the cluster prototype divergence and the cost relative to their size:

$$D^*(c_i, c_j) = D(c_i, c_j) \times \text{Merge}(c_i, c_j; \mathcal{C}) \tag{4.5}$$

where $D(.)$ is the divergence part, while Merge(.) corresponds to the size part; this ensures that quality and quantity are similar across clusters. These two parts will be defined in the following.

For simplicity, the *features* are relatively defined to the *samples* considered. When looking at rows, the features represented by columns, while when looking at columns, the relative features correspond to the rows.

4.3.2 Partitioning Entropy

4.3.2.1 Shannon Entropy

The Shannon entropy is a way to measure the number of bits required on average to code an information. The more bits are needed, the more information would transit.

For a random variable with discrete values $X = \{x_i\}$ and associated probabilities $Pr(X = x_i) = p_i$, the Shannon entropy is defined as:

$$H(X) = -\sum_{x_i \in X} p_i \log p_i \tag{4.6}$$

with the log corresponding to the base 2 logarithm $\log_2$.

4.3.2.2 Informative Clustering

We state that a partitioning $\mathcal{C}$ is informative if items are distributed into clusters of equivalent size, maximizing the entropy $H(\mathcal{C})$. Here, the probabilities associated to each cluster is not related to the item frequency, ie $Pr(C = c_i) \neq \sum_{x \in c_i} Pr(X = x)$. Otherwise, the goal would be to isolate highly frequent keywords into individual clusters and gather infrequent keywords on a large cluster, which is by no means more interesting than clustering infrequent keywords alone. Therefore, the cluster contribution is proportional to the number of items $|c_i|$, $Pr(C = c_i) = p_i = \frac{|c_i|}{\sum_{c_j \in \mathcal{C}} |c_j|}$. To make the distinction with entropy using item distribution, we refer to the entropy using item count as the *partition entropy*.

For a partitioning into k clusters, $H(\mathcal{C})$ is maximal for $Pr(C = c_i) = \frac{1}{k} \forall i$ with a maximal value of $\log k$. The partitioning with the largest entropy is the one with a single item in each cluster. To compare fairly two partitions at different stages of the agglomerative process regardless the number of clusters, the partition entropy is normalized by its theoretical maximum:

$$H_{rel}(\mathcal{C}) = \frac{H(\mathcal{C})}{\log |\mathcal{C}|} \tag{4.7}$$

which is defined for any partitioning with at least 2 clusters. The relative entropy takes values in $[0, 1]$, which allows convenient state comparison regardless the number of partitions.

4.3.2.3 Partitioning Entropy Variation

At the start, each item forms its own cluster, leading to a $H_{rel}(\mathcal{C}) = 1$. The agglomerative process leads to clusters of various sizes which affects the entropy's quality. To keep this value maximal, we study the entropy variation following a merge. For two clusters c_i and c_j merged together, with probability p_i and p_j, the new entropy can be expressed using the previous term:

$$\begin{aligned} H_{rel}(\mathcal{C}^{k-1}) &= H_{rel}(\mathcal{C}^k)\tfrac{\log k}{\log(k-1)} \\ &+\tfrac{p_i \log p_i + p_j \log p_j}{\log k} - \tfrac{(p_i+p_j)\log(p_i+p_j)}{\log(k-1)} \\ &= H_{rel}(\mathcal{C}^k)\tfrac{\log k}{\log(k-1)} + \Delta(p_i, p_j; k) \end{aligned} \tag{4.8}$$

where $\mathcal{C}^{k-1}$ corresponds to the new partition with $k-1$ clusters and $\mathcal{C}^k$ the previous partition with c_i and c_j unmerged. The total entropy is improved by a factor $\frac{\log k}{\log(k-1)}$ regardless of which clusters are merged. Concerning the merged clusters contribution $\Delta(p_i, p_j; k)$, the behavior can be estimated for large k, as the approximation $\log k \approx \log(k-1)$ holds. The merge impact is equivalent to $\Delta(p_i, p_j; k) \approx \frac{1}{\log k}\left(f(p_i) + f(p_j) - f(p_i + p_j)\right)$ where $f(x) = x \log x$. f is negative, concave and monotonically decreasing over the interval $[0, e^{-1}]$, with $e^{-1} = \exp(-1)$ corresponding to the limit of what could be considered as *small* clusters, which is respected for many partitions as $3e^{-1} > 1$. As a consequence, $f(p_i) + f(p_j) < f(p_i + p_j)$ which leads to an entropy decrease, compensated to some extend by $H_{rel}(\mathcal{C}^k)\frac{\log k}{\log(k-1)}$.

4.3.2.4 Cluster Size Influence

For large k, $\Delta(p_i, p_j) < 0$. To study the size influence, two merges are compared: the merge of c_i and c_j with respective probabilities p_i and $_j$, and the merge of c'_i with c'_j with respective probabilities αp_i and αp_j, where $\alpha \in \mathbb{R}^+$. It can be shown that $\Delta(p'_i, p'_j) = \alpha\Delta(p_i, p_j)$. The cost increases with the size of clusters merged. To maximize the relative partitioning entropy cost over the agglomerative process, small clusters must be preferentially merged to limit the loss, which can be compensated by the former term $H_{rel}(\mathcal{C}^k)\frac{\log k}{\log(k-1)}$ in eq. (4.8).

4.3.2.5 Minimization Criterion

For any pair of clusters (c_i, c_j), the term $H^{rel}(\mathcal{C})\frac{\log k}{\log(k-1)}$ in (4.8) is the same regardless of which clusters are merged. Two different merges are distinguished by the value of $\Delta(p_i, p_j; k)$. As this term is negative, the goal is to minimize:

$$\text{Merge}(c_i, c_j; \mathcal{C}) = -\Delta(p_i, p_j; k) \tag{4.9}$$

4.3.3 Content Similarity

We discussed about cluster size in the previous paragraphs. The following describes the evaluation of clusters' content similarity.

4.3.3.1 Cluster Conditional Probability

Given a cluster $c^{(X)} \in \mathcal{C}^{(X)}$ and a partitioning $\mathcal{C}^{(Y)}$, the distribution of cluster $c^{(X)}$ over $\mathcal{C}^{(Y)}$ is:

$$Pr(c^{(Y)}|\mathcal{C}^{(X)} = c^{(X)}) = \frac{\sum_{x_i \in c^{(X)}} \sum_{y_j \in c^{(Y)}} M^*_{i,j}}{\sum_{x_i \in c^{(X)}} \sum_{y_j \in Y} M^*_{i,j}} \tag{4.10}$$

and for the distribution of cluster $c^{(Y)}$ over $\mathcal{C}^{(X)}$:

$$Pr(c^{(X)}|\mathcal{C}^{(Y)} = c^{(Y)}) = \frac{\sum_{x_i \in c^{(X)}} \sum_{y_j \in c^{(Y)}} M^*_{i,j}}{\sum_{x_i \in X} \sum_{y_j \in c^{(Y)}} M^*_{i,j}} \tag{4.11}$$

4.3.3.2 Cluster Dissimilarity

For two clusters $c_a, c_b \in \mathcal{C}^{(X)}$, the probability distribution over $\mathcal{C}^{(Y)}$ is noted A and B respectively to limit notation symbols, such as $a_i = Pr(c_i^{(Y)}|c_a^{(X)})$ and $b_i = Pr(c_i^{(Y)}|c_b^{(X)})$. The Kullback-Leibler ($KL$) divergence is a way to measure the distance between probability distributions A and B:

$$KL(A\|B) = \sum_i a_i \log \frac{a_i}{b_i} \tag{4.12}$$

The same equation is obtained for $c_a, c_b \in \mathcal{C}^{(Y)}$, with $a_i = Pr(c_i^{(X)}|c_a^{(Y)})$ in this case. The intuition of the KL divergence is that mass of distribution B must be present where A is. If not, the penalty grows. The KL divergence can be rewritten as $KL(A\|B) = H^*(A,B) - H(A)$ where $H^*(A,B)$ is the cross-entropy, and $H(A)$ the regular entropy. In other words, KL represent the extra-cost of coding A using B's code.

4.3.3.3 Symmetry

There is no order when merging two clusters, as $c_i \cup c_j = c_j \cup c_i$. However, the Kullback-Leibler divergence is not symmetric. Instead, we use the J-symmetrized KL divergence, which is defined as:

$$KL^J_\alpha(A\|B) = (1-\alpha)KL(A\|B) + \alpha KL(B\|A) \tag{4.13}$$

with the balance factor $\alpha = \frac{1}{2}$. The symmetrization ensures that both A and B share the same support probability, which leads to a more discriminative function as both a_i and b_i needs to be non-zero for the same feature i.

4.3.3.4 Minimization Criterion

The cost term takes into account the clusters similarity in eq. (4.5) is $D(c_a, c_b) = KL^J(A\|B)$. It measures the divergence between prototypes' distribution according to the clustered features. The KL^J takes values in $\mathbb{R}^+$, where a low value represents a high content similarity between considered clusters.

4.3.4 Agglomeration Procedure

Given the initial smoothed documents-keywords matrix M^*, the algorithm starts by computing KL^J for all possible pairs of clusters in $\mathcal{C}^{(X)}$ and pairs in $\mathcal{C}^{(Y)}$. This lead to two initial divergence matrices $KL^J(\mathcal{C}^{(X)})$ and $KL^J(\mathcal{C}^{(Y)})$. This operation is computationally expensive, as it requires $\mathcal{O}(nm(m+n))$ operations.

The merge cost is recomputed at each round, and the total cost is computed for each pair. The pair of clusters from $\mathcal{C}^{(X)}$ or $\mathcal{C}^{(Y)}$ with the lowest cost is selected and merged. The two divergence matrices are updated after the merge operation. If two clusters $c_i, c_j \in \mathcal{C}^{(X)}$ are merged together, all the pairs involving c_i and c_j in $KL^J(\mathcal{C}^{(X)})$ must be recomputed with the new cluster characteristics $c_i \cup c_j$, leading to a matrix with one dimension less. This first update requires $\mathcal{O}(n(t)m(t))$ operations, where $n(t) = |\mathcal{C}^{(X)}|$ and $m(t) = |\mathcal{C}^{(Y)}|$ is the number of row and column clusters left after t merge operations.

Concerning $KL^J(\mathcal{C}^{(Y)})$, all the items are affected. Nonetheless, the matrix can easily be updated, by looking at the difference between merged and unmerged state. For two column clusters c_a and $c_b \in \mathcal{C}^{(Y)}$ with distribution over rows A and B respectively, the cost variation is:

$$\begin{aligned} \Delta^{(i,j)}KL(A\|B) &= (a_i + a_j)\log\tfrac{a_i+a_j}{b_i+b_j} \\ &\quad - \left(a_i \log\tfrac{a_i}{b_i} + a_j \log\tfrac{a_j}{b_j}\right) \end{aligned} \tag{4.14}$$

where $\Delta^{(i,j)}KL(A\|B)$ corresponds to the non-symmetric KL cost. The new cost for merging c_a with c_b is replaced by $KL^J(c_a, c_b) + \Delta^{(i,j)}KL(A\|B) + \Delta^{(i,j)}KL(A\|B)^\dagger$. This updating step requires $m(t)^2$ operations to update all the feature pairs. In total, a step requires $\mathcal{O}(n(t)^2 + m(t)^2)$ operations to select the best pair, and $\mathcal{O}(n(t)m(t) + m(t)^2)$ operations to update the cost pairs KL^J when merging two row clusters. The same reasoning applies when merging two-column clusters by exchanging $n(t)$ with $m(t)$ in the formula.

4.4 Experimental Setup

4.4.1 Datasets

4.4.1.1 Scientific Paper Tags

The main motivation for tag co-clustering arose from scientific papers literature. The DBLP dataset [27] is a citation graph gathering computer science papers, with meta-data such as title, publication year, references, authors, conference/journal, *field of study*, available for a large number of papers. We used the most recent version (v12) for our experiments.

The *field of study* is a list of descriptive keywords about the field (e.g. *Cryptography*, *Biology*), the method (*Matrix factorization*), or other related concepts (*Bullwhip effect*) discussed in a paper. Each paper has, on average 10 descriptive tags. Hopefully, tags are already well pre-processed, and no steaming nor stop-words removal need to be done.

The algorithm complexity is more than quadratic, which prevents the scaling to a large database. Documents are sampled to make the clustering possible on a regular machine. A particular tag is selected, and all papers with the tag included are gathered. Then, 5000 documents are selected at random from this pre-selection. All keywords with less than 5 occurrences are discarded, which leads to around 1000 keywords left and a filling rate of 2% of the binary documents-keywords matrix.

4.4.1.2 Synthetic checkerboard

The real-world dataset does not contain any label, which prevents the evaluation with objective metrics. We propose to generate a sparse synthetic dataset with clustering structures to test the performance of our model.

A synthetic sparse matrix M of size $n_X \times n_Y$ partitioned over X and Y dimensions is constructed the following way. Rows are split into k_X clusters of equal size $\lfloor \frac{n_X}{k_X} \rfloor + \{0, 1\}$. The same regular partitioning is performed on Y with k_Y clusters.

The matrix M is filled with 0 and 1 according to the partitioning. The *tile* $T(a, b) = \{M^*_{i,j}\}_{x_i \in c_a^{(X)}, y_j \in c_b^{(Y)}}$ is the intersection between the row and column clusters $c_a^{(X)}$ and $c_b^{(Y)}$. Some of the tiles selected with probability $\alpha \in [0, 1]$ are filled, leaving $(1 - \alpha)$ of the tiles empty, where α is the *global* filling rate. For a tile $T(a, b)$ to fill, the filling rate $\beta_{a,b}$ is selected at random in $[0, \beta]$ with $\beta \in [0, 1]$ the *local* filling rate. For each item (i, j) in tile (a, b) to be filled, its value is 1 with probability $\beta_{a,b}$ else 0. The result is a matrix filled with rate $\frac{\alpha\beta}{2}$.

In our case, the global and local filling rate are set to $\alpha = \beta = 0.2$ leading to a total filling rate similar to our real-world dataset of 2% . For all experiments, $n_X = n_Y = 1000$ and $k_X = k_Y$ would be adjusted over the experiments. The resulting matrix looks like a regular grid and would be called, for this reason, the *checkerboard* dataset (see Fig. 4.1). The experiments are done for the same n_X and n_Y, and identical k_X and k_Y. This choice is made to aggregate

results over X and Y together, but the performances are not affected by asymmetric choices.

4.4.2 Monitoring metrics

To evaluate our algorithm, we selected some supervised and unsupervised metrics to evaluate the quality of the partition recovery and estimate cluster quality in the absence of labels.

4.4.2.1 V-measure

The V-measure is a supervised metric comparing the real clusters to the estimated ones using entropy measures. It is analogous to *accuracy* on classification problems. Two sub-measures are first computed: the *homogeneity*, which corresponds to the fact that a good cluster contains a single class, and the *completeness*, which measures how well elements from a given class are grouped. The homogeneity is defined as:

$$h = 1 - \frac{H(L|K)}{H(L)} \tag{4.15}$$

with L the real cluster labels and K the hypothetic labels obtained using a clustering. The completeness is defined similarly as:

$$c = 1 - \frac{H(K|L)}{H(K)} \tag{4.16}$$

The V-measure is then defined as:

$$V = \frac{(1+\beta) \times hc}{\beta h + c} \tag{4.17}$$

where the parameter $\beta \in \mathbb{R}^+$ balances the contribution of each term. For $\beta = 0$, it corresponds to the h, while $\lim_{\beta \to \infty} = c$. The values obtained lie within $[0, 1]$, where 1 is attributed to the best clustering, while 0 to the worse case. The V-measure can also be used to compare two partitioning obtained with different parameters or algorithms, measuring the similarity degree.

Random Guess Suppose $\mathcal{C}$ is a partitioning of items into k clusters of equal size, with $Pr(c) = \frac{1}{k}$. The associated partitioning entropy is $H(\mathcal{C}) = \log k$. Be $\mathcal{C}'$ a randomly guessed partitioning with k clusters of equal size too, but filled with items selected at random. The overlap probability between $c \in \mathcal{C}$ and $c' \in \mathcal{C}'$ is $\frac{1}{k^2}$ for all clusters' pairs. Consequently, the joint entropy is $H(\mathcal{C}, \mathcal{C}') = \log k^2 = 2 \log k$, leading to $H(\mathcal{C}|\mathcal{C}') = H(\mathcal{C}'|\ \mathcal{C}) = \log k$. Completeness and homogeneity are both equal to 0, leading to an undefined V-measure. However, when looking at the limit, the value converges to zero. Compared to accuracy measure, where a random guess's accuracy is $\frac{1}{k}$, the V-measure is more discriminative.

4.4.2.2 Limited Partitioning Entropy

At the start, the partitioning entropy defined in eq. (4.7) is maximal. However, the clustering is non-informative as only singleton clusters exist. Instead, knowing that final clusters would have a critical size with more than r elements, smaller clusters' contribution can be discarded, considering them as *outliers*. The restricted relative partitioning entropy is then defined as:

$$H^*_{rel}(\mathcal{C}; r) = \frac{-\sum_{c \in \mathcal{C} \wedge |c| > r} p(c) \log p(c)}{\log |\mathcal{C}|} \tag{4.18}$$

At initialization, the value is 0 as no cluster of sufficient size exists for $r > 1$. This value is still bounded between $[0, 1]$ and enables to track clusters creation. This measure allows to evaluate the partitioning distribution without considering cluster content.

4.4.2.3 Mutual Information

When monitoring the cluster's content, the information variations are monitored. In this case, the entropy is computed using the sample probabilities, defined in equations (4.10) and (4.11). The *mutual information* corresponds to the information shared between X and Y. This measure is defined as:

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \tag{4.19}$$

It corresponds to the gain of coding X with Y, i.e. the amount of redundancy between the two variables. This value is bounded by 0 (independent variables) and $\min(H(X), H(Y))$ (correlated variables). We would monitor the mutual information between partitioning $I(\mathcal{C}^{(X)}, \mathcal{C}^{(Y)})$. As the number of clusters decrease over time, this value would decrease due to information loss during the compression.

4.4.3 Comparative Algorithms

We proposed to compare our algorithm to the co-clustering algorithm presented in [63], which relies on spectral decomposition. First, two diagonal matrices D_1 and D_2 are obtained from M, where $D_{1:i,i} = \sum_{j=1}^{m} M_{i,j}$ and $D_{2:j,j} = \sum_{i=1}^{n} M_{i,j}$. Then, the normalized matrix $M_n = D_1^{-1/2} M D_2^{-1/2}$ is decomposed using singular value decomposition such as $M_n = USV^T$. The vectors $D_1^{-1/2}U$ and $D_2^{-1/2}V$ are concatenated to form the matrix Z. The algorithm finishes by performing a k-means clustering on the $\ell = \lceil \log_2 k \rceil$ main dimensions, omitting the first main dimension.

4.5 Experimental Results

4.5.1 Smoothing Effect

The smoothing proposed enables to switch from binary values to real values by redistributing the weights. A visual example is presented in Fig 4.1. The filled tiles are identifiable on the binary matrix. For the tiles with low $\beta_{a,b}$, the boundaries are hard to identify.

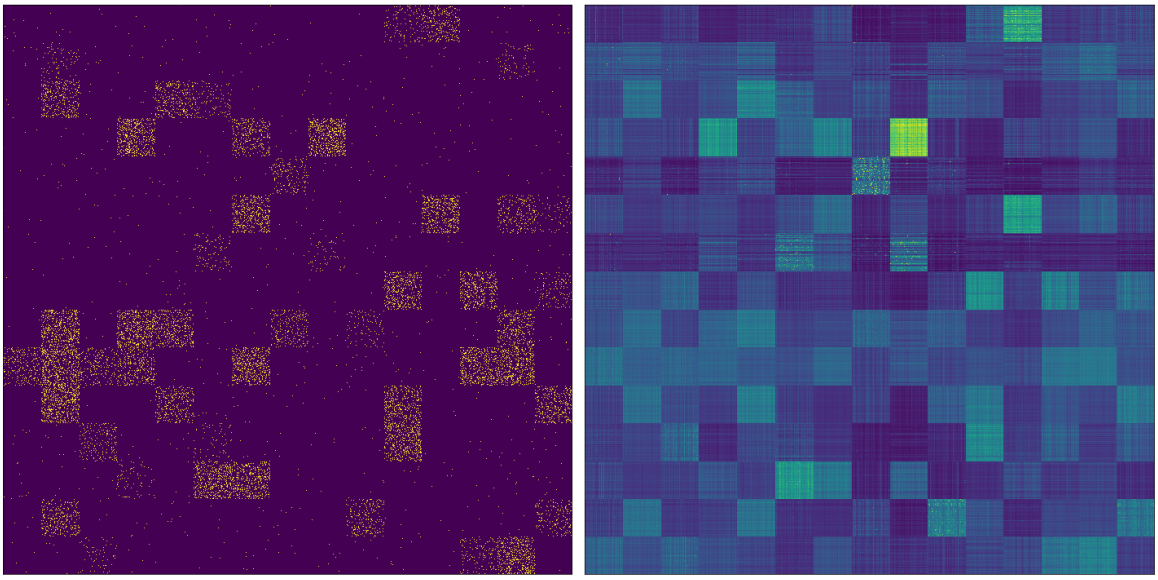


Figure 4.1: Matrix smoothing for $n_X = n_Y = 1000$ and $k_X = k_Y = 15$, ie around 67 points per cluster. The global filling ratio is $\alpha = 0.2$ and the local filling ratio is $\beta = 0.2$. Left: binary matrix, right: smoothed matrix.

On the smoothed matrix, the weights are completely redistributed leading to the visual identification of tiles, even the ones that were not filled at all. All items belonging to the same clusters tend to have a more similar feature vector. The information is of lower intensity locally but is better distributed across the different features, even on tiles that were not filled.

4.5.2 Size Dependent Cost

This experiment compares the composite cost defined in eq. (4.5) to the version where only content similarity obtained using KL^J cost is taken into account. For this purpose, we compared for several numbers of cluster k the maximal V-measure averaged over 5 independent trials for each k. We gathered the number of clusters left $\hat{k}$ for the maximal value of V.

We did the same by extracting the maximal restricted relative entropy, removing clusters of size smaller or equal to 1, and extracting the corresponding $\hat{k}$.

The results are presented in Fig. 4.2, with the composite cost denoted $KL \times H(\mathcal{C})$ and the simple cost KL.

As a general remark, for all setups, it is easier to cluster many clusters with sufficient size. For very few clusters, the accuracy quickly decreases. As the global filling rate is $\alpha = 0.2$, on

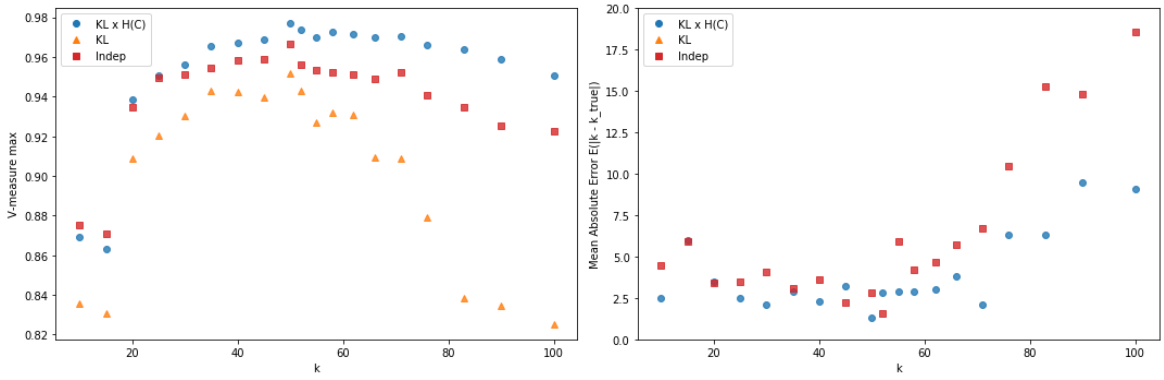


Figure 4.2: Best v-measure and average absolute error for corresponding k. Blue rounds correspond to the normal setup with co-clustering and weighting according to KL^J cost and cluster distribution entropy $H(\mathcal{C})$. Orange triangles correspond to the co-clustering setup without taking into account cluster distribution. Red squares correspond to the double weighting approach but rows and columns are clustered independently. Each point corresponds to the average for 5 independent trials.

average 2 tiles are filled over 10. Due to randomness, a single one or none could be filled, leading to less information for clustering. For a large number of partition, a cluster is well defined in the sense that enough tile are filled. The problem of insufficient information occurs at the sample level. For a row sample of size n and k feature partitioning, there are $\frac{n}{k}$ slot for a given feature cluster. As the local filling rate is $\beta_{a,b} \leq 0.2$, the probability that none of the slot are filled grows with k. This effect is nonetheless less disturbing than the former as redundancy exists.

The V-measure of the composite cost is always higher than for the simple cost setup. The accuracy of KL quickly drops for large k with smaller cluster sizes. The absolute k deviation $\mathbb{E}(|k - \hat{k}|)$ of the composite cost is relatively close from the optimal, with a consistent error unless for very small cluster size, while the error for the simple cost is too large to fit in the figure.

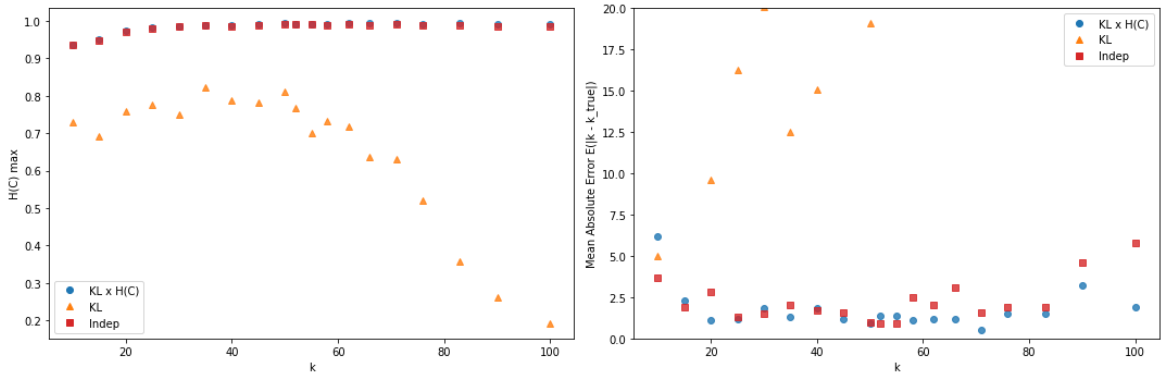


Figure 4.3: Left: Best $H(\mathcal{C})$ with cluster of size lower or equal to 1 removed. Right: average absolute error of the corresponding $\hat{k}$. Same color code and setup as 4.2. Blue rounds and red circles overlap on the left sub-figure.

Looking at the maximal relative partitioning entropy in Fig. 4.3, the composite cost leads to very good cluster distributions for any k, while the simple cost is 0.2 points lower for small number of clusters and very low for larger values. On the right side of Fig. 4.3, the $\hat{k}$

obtained with the composite cost is close to the exact value even for large k. This means that the number of clusters obtained when stopping the agglomeration procedure with the partitioning entropy criterion is close from the true initial cluster numbers for all cluster sizes. When using the simple cost, clusters number is far from the true number of cluster. In both case, the entropy stopping criterion leads to a smaller error over the number of estimated clusters.

4.5.3 Co-Clustering vs Independent Clustering

One of the initial hypothesis concerns the synergy between joint reduction. We compare the co-clustering setup to the independent setup, where the partitioning $\mathcal{C}^{(X)}$ is obtained using the uncompressed features Y, as well as the partitioning $\mathcal{C}^{(Y)}$ is obtained using the unaggregated rows X. This setup is denoted *Indep* in Fig. 4.2 and 4.3.

The results obtained with the independent clustering are similar to the co-clustering but with a lower v-measure for a large number of clusters k. The obtained $\hat{k}$ from the v-measure are close to the co-clustering ones. As far as shape is concerned, independent clustering performs as well as the co-clustering, and the $\hat{k}$ obtained are relatively similar. The co-clustering advantage is limited for large clusters / small k and becomes more interesting when uncertainty grows with smaller size clusters for large k.

4.5.4 Comparison to Alternative Algorithms

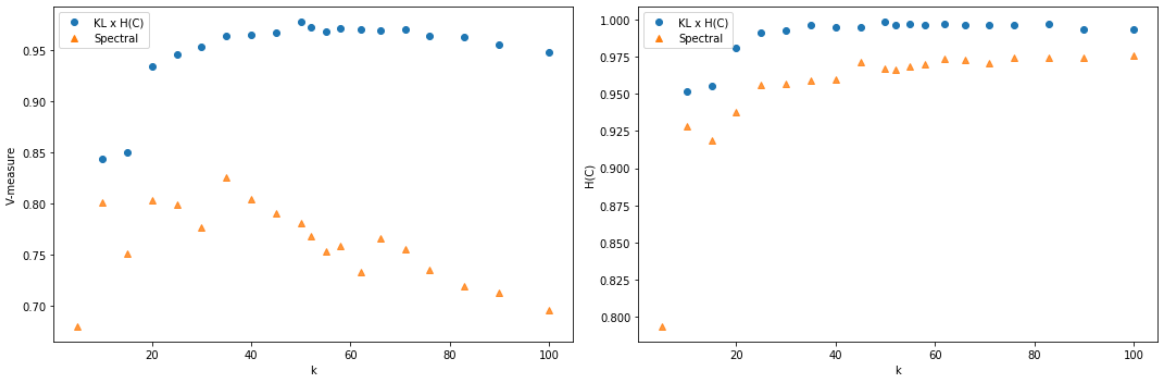


Figure 4.4: Comparative results between Spectral co-clustering and Agglomerative clustering. On the right, maximal V-measure obtained for

We compare our algorithm to spectral co-clustering presented in [63]. The algorithm needs as input the target number of clusters to search for. When comparing the agglomerative algorithm with the spectral algorithm, the spectral algorithm is run with the exact k provided, compared to the partitioning obtained with the agglomerative algorithm with k remaining clusters. The V-measure and the relative cluster partitioning entropy $H^*_{rel}(\mathcal{C}; 1)$ are extracted from these two partitioning. The results are presented in Fig. 4.4.

The spectral algorithm results are lower than the one obtained for the agglomerative approach. However, the shape of the clusters obtained are equivalent. The spectral approach is quite

robust in general, but the sparsity level affects the results. With a higher filling rate ($\alpha = 0.4$), the spectral results get closer to the agglomerative one.

We also tested with DBSCAN, which has been used in some papers. As it is impossible to select the wished number of clusters, and because the results were lower than spectral decomposition, the results are not presented. Nonetheless, the smoothed matrix's use improved the partitioning, allowing the algorithm to discover more clusters than with the regular binary matrix.

4.5.5 Textual Results

The initial goal was to cluster tags associated with scientific papers to identify topics. In the dataset used, there is no high-level classification or paper grouping to evaluate our clustering. Despite the lack of objectivity, we present the results on two subsets of papers, obtained for the *Payment* field of study, and the second for *Biometry*.

We take advantage of the hierarchical form to present the results using a dendrogram. Around 15 clusters are left unmerged, and the three most frequent keywords are displayed for analysis. For the two, the relative partitioning entropy was around $H(\mathcal{C}) = 0.95$ for keywords.

Payment Fig. 4.5 corresponds to the clustering of keywords co-occurring with the *Payment* tag. Three high level clusters are identified. The one on the left corresponds to things related to *economics*. The right one corresponds to what could be considered as the *core* of the payment field, oriented toward users, with the new payment methods (*Cryptocurrency*, *Mobile payment*) and intricated topics (*Computer security*, *Marketing* and *Advertising*). The bottom cluster corresponds to the medium or technology used in the payment but is not specific. For instance, *Artificial intelligence* is used in payment systems for fraud detection or biometric authentication, but it is not specific to payment.

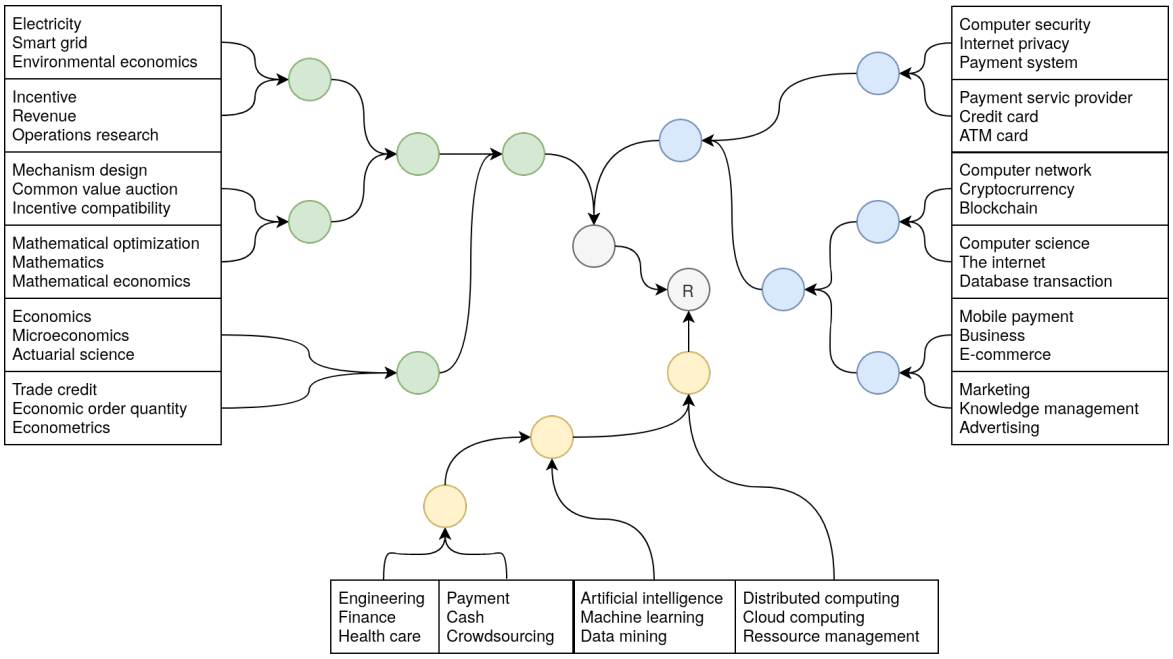


Figure 4.5: Dendrogram for *Payment* field of study.

Surprisingly, the keyword *Payment* is located on the bottom cluster, near *Cash* and *Crowdsourcing*, which seems conceptually incorrect. This is due partially to our sampling method, where all documents with keywords *Payment* were selected. As it co-occurs with all keywords, there is no way to identify true relationships. *Payment* is located on a cluster were the other keywords are related to *Crowds*, with additional keywords such as *Reputation*, *Social network*, *Audit* and *Crowdsensing*.

This artefact is not limited to the selected keywords but to the most frequent keywords. A second example is *Computer science* on the right, in a cluster related to the *Internet*, with additional keywords like the *World Wide Web*, *Mobile device*, *Service provider* and *Mobile computing*.

Biometry The second partitioning uses the *Biometrics* tag as a reference. The resulting dendrogram is presented in Fig. 4.6. Two large clusters are identified. The main on the left gathered keywords about biometric methods and algorithms to extract a digital identity. It is subdivided into two subgroups.

The top one gathered keywords related to computer-vision, with *Image processing*, gait and face analysis. The bottom one corresponds to the other methods, with fingerprint, *Speaker recognition/verification*. The cluster with *Biometrics*, *Computer science* and *Speech recognition* corresponds to an artefact gathering highly frequent keywords together. The right cluster corresponds to the security part, with *Cryptography*, *Password*, *Authentication* and others.

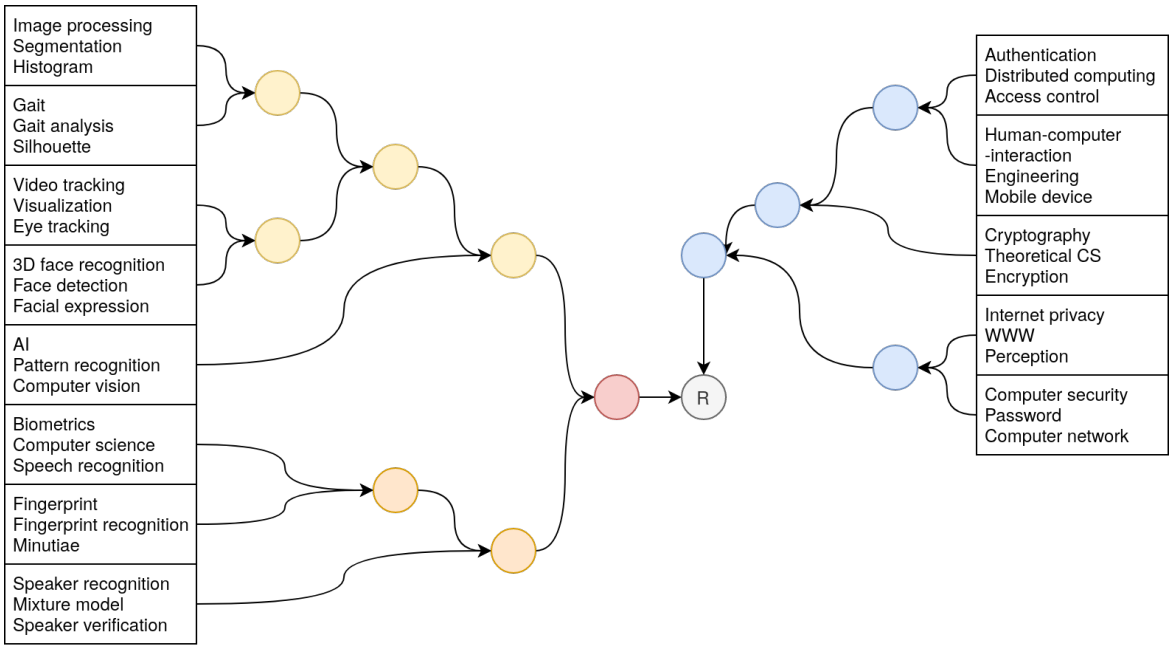


Figure 4.6: Dendrogram for *Biometrics* field of study.

4.6 Discussions

4.6.1 Ending Criterion

The checkerboard experiments were evaluated, knowing the number of clusters. In an unsupervised setup, this knowledge is often unavailable. To select the cluster number, one has to look at a specific criterion indicating if the partitioning is satisfying. For instance, the algorithm X-means [69] is a divisive algorithm based on k-means which successively splits the

existing clusters. The splitting decision is based on the split's likelihood, assuming the data corresponds to a Gaussian mixture. For more general clustering algorithm, the silhouette coefficient, measuring the distance to the nearest cluster versus the radius of the cluster.

On our type of data, the silhouette is not suitable as cluster are not well separated. The *goodness* criterion of the algorithm must be in accordance to the goal achieved by the algorithm. Reminding our cost definition in (4.5), it is the product between *cluster size* and *content* related costs.

The first part of the answer to this problem is to look at the restricted relative partitioning entropy defined in (4.18), with small clusters of size 1 or less removed. In Fig. 4.3, $H_{rel}(\mathcal{C}; 2)$ is already a good indicator of when clusters are sufficiently aggregated. However, this is a particular case where all clusters have the same size, leading to a particular configuration where H_{rel} is maximal. In a more general configuration, there is no particular reason for clusters of exactly the same size.

The second cost part takes into account content. On the information theory-based work of [64], a good clustering is defined as minimizing the quantity $I(X, Y) - I(\mathcal{C}^{(X)}, \mathcal{C}^{(Y)})$ for a given number of row and column clusters. The agglomeration of clusters is a form of compression which mechanically reduces the information available.

The restricted relative entropy is maximal towards the end of the agglomeration process, while the information is maximal at the beginning and minimal at the end. A good compromise between the two is to look for the value for which the product of $I(\mathcal{C}^{(X)}, \mathcal{C}^{(Y)})$ and $H(\mathcal{C}^{(X)})$ or $H(\mathcal{C}^{(Y)})$ is maximal:

$$k_X^* = \arg_{k_X = |\mathcal{C}|} \max H_{rel}(\mathcal{C}^{(X)}; r) \times I(\mathcal{C}^X, \mathcal{C}^Y) \tag{4.20}$$

X's best partitioning is not necessarily simultaneous with that of Y because the actual number of clusters may be different. The co-clustering only exploit synergies to find clusters more accurately. In general, the partitioning with the lowest number of dimensions would be merged more frequently until reaching a size comparable to its feature size. As a rough guide, for k features, the maximum entropy is $\log k$. The cost of KL^J is not limited by an upper bound, but the higher the number of features, the higher the cost will be because the probability of $a_i = 0 \neq b_i$ is higher in such a configuration. With a higher cost, the clustering will preferably select the cluster pairs with the smallest number of features.

This criterion was tested on the checkerboard dataset. The estimated $\hat{k}$ were very close to the expected one, with an average absolute deviation close to 1. This criterion was used to build the dendrogram, where the estimated cluster number was between $15 \sim 20$ clusters depending on the main selected keyword.

4.6.2 Model Limitations

The checkerboard model differs from a documents-keywords matrix obtained from tag on two majors points.

The first difference concerns distribution. Tags follow a Zipf's law, which is not modeled here, as all columns have on average the same strength. Nonetheless, the unbalanced distribution is corrected by the matrix smoothing protocol, which decreases the weight of these frequent keywords to less frequent one.

The second difference is the hierarchical division of keywords. The associated tags range from very broad domains (*Chemistry*, *Mathematics*), to fields (*Inorganic chemistry*, *Databases*), to specialities and other lower levels. The checkerboard model is made of independent clusters which are not hierarchicaly organized. Nonetheless, due to the scaling limitation of the proposed algorithm, the restriction to around 5000 documents and 1000 keywords limit the visibility of such organizations.

4.7 Conclusion

In this paper, we addressed the problem of tag clustering, where the tag amount per document is limited. To this purpose, using the correlation between tags and keywords, a method to enhance context without the use of an external database or model was proposed. With the assumption that a clustering is informative if the partition entropy is large, we proposed an agglomerative co-clustering algorithm taking into account the content as well as the cluster size. The algorithm showed good recovery performance on synthetic datasets with the same sparsity level. It showed conceptually correct clustering results on scientific paper tags, up to highly frequent keywords where no discriminative relationship could be found. The algorithm's complexity is polynomial but more than quadratic, which restricts its usage on a small dataset. Some improvement can be made by splitting the dataset into independent parts or finding cost approximations. Nonetheless, the idea of building groups of equivalent size could be mixed with other agglomerative measures to include distant items to their closest cluster.

CHAPTER 5

Co-Embedding Bipartite Graphs

Many datasets take the form of a bipartite graph where two types of nodes are connected by relationships, like the movies watched by a user or the tags associated with a file. The partitioning of the bipartite graph could be used to fasten recommender systems, or reduce the information retrieval system's index size, by identifying groups of items with similar properties.

This type of graph is often processed by algorithms using the Vector Space Model representation, where a binary vector represents an item with 0 *and* 1*. The main problem with this representation is the dimension relatedness, like words' synonymity, which is not considered.*

This article proposes a co-clustering algorithm using items projection, allowing the measurement of features similarity. We evaluated our algorithm on a cluster retrieval task. Over various datasets, our algorithm produced well balanced clusters with coherent items in, leading to high retrieval scores on this task.

5.1 Introduction

Many datasets can be represented as a bipartite graph (BPG), making the links between two different types of items. This could be the movies a user watched, the purchases he made, the music he listened at. Outside of user interaction, it could be tags associated with a file, such as the keywords of an article, or the genes with which a molecule interacts.

Community discovery or graph partitioning helps improving scalability in different contexts. Collaborative Filtering Recommender Systems (CF RecSys) work by suggesting new items to a user based on the similarity of its history to the other users' history. Without optimization, a user is compared to all the other users to find those most similar to him. Clustering reduces costs by looking at similar users within the same partition rather than looking at them in the entire dataset [70]. Clustering is not limited to users and has similar benefits when clustering items [71], or both together [72]. These approaches are based on the clustering hypothesis [73] supposing that similar documents would answer the same information needs. Cluster-based retrieval systems are based on this same hypothesis [74,75]. A very large database that cannot be stored in a single server could be split into multiple servers, hosting a particular thematic cluster. A query would be compared to an index with clusters' summary and routed to the most relevant server, reducing the overall number of operations.

There are different approaches to cluster a bipartite graph. The algorithms can be classified into two distinct categories: *one-way* and *two-way* partitioning algorithms, with the latter grouping both sides of the graph simultaneously. In the *one-way* approach, the set of nodes belonging to the same type are clustered, without considering the clusters that would be obtained when clustering the nodes from the other type. The nodes to cluster are often represented into a matrix using the Vector Space Model (VSM) [76], where the nodes of the second type are considered as their features [77] or used to construct the features [78]. The resulting matrix can be exploited to cluster items with usual algorithms such as k-means, measure the items' proximity using cosine similarity measure, or project items using dimensionality reduction algorithms [78].

Another possibility is to convert the 2-mode graph into a 1-mode graph by collapsing the graph. The conversion to a unipartite graph is convenient as many classical community detection algorithms can be used [79]. The nodes from a single type are preserved and new weighted edges are inferred from the bipartite graph. There are multiple approaches to convert the bipartite graph into a unipartite graph [80,81]. However, the main problem is the growing number of edges and the loss of information due to the collapse [79,80].

In contrast, *two-way* clustering algorithms – also called co-clustering or simultaneous clustering [82] – exploit the raw structure to cluster both types of nodes simultaneously. *Two-way* approaches lead to better results, even if the goal is to partition only one side of the BPG [64,82] by exploiting synergies between the obtained clusters.

Among co-clustering approaches, the majority of the literature focuses on *bi-clustering*

approaches, where a bicluster is characterized by a particular set of rows and columns when the BPG is represented under the matrix form. Therefore, a bicluster gathers items of heterogeneous type. There exist sub-categories characterized by the possibility to overlap clusters and to assign an item to multiple clusters [83]. The majority of the algorithms creates biclusters with exclusive rows and columns [79,84,85]. In this situation, the models assume a one-to-one match between sample and feature clusters and strong connectivity, forming blocks when visualizing the matrix with items grouped by clusters. This clustering type is – to some extent – equivalent to finding sub-graphs forming dense structures [79] that could exist without the other groups.

In contrast, the literature covering the case where a cluster gathers items of the same type is limited. While this case seems to correspond to the one-way partitioning, it highly differs from it as a row (column) cluster is expressed as a mixture of column (row) clusters. In contrast, the one-way approach considers features individually. Among this group, we can mention non-negative matrix factorization (NMF) approaches [86,87]. This approach offers a more flexible framework than the latent block model leading to a checkerboard representation when re-ordering rows and columns by groups. Nonetheless, this approach assumes that there are latent variables connecting rows to columns; therefore we obtain the same number of row and column clusters.

In this work, we particularly focus on the case where the number of row and column clusters are not necessarily equal. These approaches are more flexible as the number of clusters is not constrained by the reciprocal type of items and increase the number of possible configurations. In this group, we can mention [63] combining a spectral decomposition to a k-means, and [64] based on information theory minimizing the loss of mutual information between clustered and unclustered items. These models require as input the number of clusters to find, limiting their usability without prior knowledge. The work of [88] extends the latent block model to cluster heterogeneous data types (ordinal, continuous, count). Samples are clustered into groups without specific constraints, while features are clustered with features of the same type. The number of feature clusters is automatically inferred and independent from the number of sample clusters, leading to a flexible automatic co-clustering approach.

Most of these approaches are designed for very general cases and do not assume the underlying nature of BPG. More specifically, they do not consider *dimension relatedness*. If animals are features like [tiger, lion, frog], the distance between their respective binary vectors $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ is 2 for all possible pairs despite the similarity between *tiger* and *lion*. The different features are considered as orthogonal, while it is far from the reality where some characteristics are often correlated, even weakly.

Dimension relatedness has been addressed mainly for textual applications. The works of [89,90] propose using an external database (WordNet and Wikipedia resp.) to measure dimension relatedness. Over a very large corpus, the dimension correlation could be learned, as proposed in the work of [91]. However, an external database is not always available,

depending on the language used, and the type of objects considered. The work of [92] suggest learning the similarities between features by directly measuring their textual similarity. Feature similarity is derived from the Levenshtein distance between the feature names.

These different textual approaches learn the distances between features, but none between samples. One would like to apply the same treatment to samples and features in a co-clustering approach exploiting the sample-feature duality.

In this article, we propose a co-clustering algorithm that addresses the problem of dimension relatedness. The proposed approach follows a co-embedding process, where each side of the bipartite graph is projected in a low dimensional space. This projection enables to measure items relatedness based on their features' location. The process alternates between projecting each side of the graph, until invariance compared to the previous embedding.

The rest of this article is organized as follows. First, the algorithm is detailed in the following section, then the datasets and evaluation methods are presented. Visual, numerical, and textual results are then presented in the experimental section, followed by possible extensions in the discussion and a conclusion.

5.2 Co-Embedding

Notations: A bipartite graph $\mathcal{G} = (V^I, V^{II}, E)$ is a graph composed of two types of nodes $V^I = \{v_i^I\}_{i=1:|V^I|}$ and $V^{II} = \{v_i^{II}\}_{i=1:|V^{II}|}$. The edges E connecting nodes only exist between nodes of different types $E \subseteq V^I \times V^{II}$. The existence of a link between two nodes $v^I \in V^I$ and $v^{II} \in V^{II}$ is denoted as $\delta(v^I, v^{II}) = 1$ if $(v^I, v^{II}) \in E$ else 0. The representation of $\mathcal{G}$ under the VSM model is $M = [\delta(v^I, v^{II})]_{v^I \in V^I, v^{II} \in V^{II}}$ The number of occurrences of $v^I \in V^I$ is denoted $|v^I| = \sum_{v_i^{II} \in V^{II}} \delta(v^I, v_i^{II})$ and defined similarly for a node of V^{II}.

Sample-Feature Duality: A *feature* is relative to the *sample*'s type considered. Nodes of V^I are features of V^{II} and V^{II} of V^I. Therefore, the equations would be written in terms of samples $\mathcal{S}$ and features $\mathcal{F}$ instead of V^I and V^{II}, as our process inverses the roles periodically. As most of the datasets used correspond to (tag, resources) pairs, we would refer to *tags* $\mathcal{T}$ and *resources* $\mathcal{R}$ when a difference needs to be underlined.

Proposed Approach: We address the problem of dimension relatedness for co-clustering a bipartite graph by projecting the items into a low dimensional space. The samples relatedness is measured by comparing their features' location on the embedding space. Based on their similarity, samples are next embedded into another low dimensional space using the t-SNE algorithm [28] which is a key point in the process. Next, samples and features exchange their role to start a new iteration. The process is repeated several times until the neighborhood around each item is stable. The last step of our method is an automated clustering using the Mean-Shift algorithm [29] over the two last co-embedding.

The details of each step will be detailed in this section. First, we start with the t-SNE algorithm to provide an understanding of the embedding properties. Next, we detail how features' relatedness is measured and used to measure samples similarity. Last, we explain the procedure for Mean-Shift clustering concluding the co-clustering process.

5.2.1 t-SNE Embedding

These paragraphs describe the general ideas behind the t-SNE algorithm [28]. For a set of n items X, this non-parametric embedding algorithm transforms it into a low dimensional representation $Y \in \mathbb{R}^{n \times d}$, with d often set to 2 for visualization purposes:

$$Y \leftarrow t\text{-SNE}(D(X); d, perp) \tag{5.1}$$

The *perplexity* parameter (*perp*) controls the number of nearest neighbors in X to preserve in Y. This prevents highly connected items of the graph from having too many neighbors and improves the neighborhood of weakly connected items. The algorithm works by minimizing the Kullback-Leibler divergence between the image matrices of X and Y, obtained respectively using a Gaussian kernel and a t-Student kernel. This kernel asymmetry creates repulsive long-range forces leading to well-separated groups. Another characteristic is the homogeneous scaling over an embedding, allowing to measure similarity the same way independently of the location and the crowdedness.

The perplexity governs the embedding shape, where a large value focuses on large scale structures, while a lower one on details. The understanding of *large* is relative to the number of items, and could be adapted to tags and resources with $perp^{\mathcal{R}}$ and $perp^{\mathcal{T}}$ as they do not necessarily have the same size.

As this algorithm tries to preserve local neighborhood relationships, nothing can be said on long-range distances. However, the proximity between items in the embedding can be exploited as an alternative to distances in the high dimensional space.

5.2.2 Representing Samples using their Features Embedding

The binary vector of a sample s is transformed into a vector of probabilities. The feature embedding $Y^{\mathcal{F}} = \{\mathbf{y}_i\}_{i=1:|\mathcal{F}|}$ is used to create this vector by taking into account three factors: the feature's location, the feature's popularity, and the related features. The vector representing s is $\mathbf{p}(s) = [p(f|s)]_{f \in \mathcal{F}}$, where the term concerning a feature f is defined as:

$$p(f|s) = \frac{1}{c(s)} \sum_{f_i \in \mathcal{F}} \frac{\delta(s, f_i)}{|f_i|} K(\mathbf{y}, \mathbf{y}_i) \tag{5.2}$$

where $\frac{\delta(s,f_i)}{|f_i|}$ represents the contribution of feature f_i contained in s. The normalization constant $c(s)$ ensures that $\sum_{f \in \mathcal{F}} p(f|s) = 1$. For a feature f_i of s with image $\mathbf{y}_i$, the kernel redistributes the feature's mass on the neighborhood feature f with image $\mathbf{y}$ based on their

kernelized distance $K(\mathbf{y}, \mathbf{y}_j)$. Therefore, the weight $p(f|s)$ can be non-zero even if f is not a feature of s. *Mass redistribution* assumes that all existing edges are *true* edges and some edges are missing, i.e. there are no misconnected items, but the information available is incomplete. The kernel allows us to consider the unlinked items with some degree of confidence based on their proximity.

Kernel Choice The t-SNE embedding uses a t-Student kernel to map items. While it seems a natural kernel choice, this distribution has a long tail, allowing distant items to contribute. In the t-SNE algorithm, this kernel was chosen to create long-range forces for better clusters' separation. As the goal is to identify closely related items, the use of a Gaussian kernel is more adapted, defined as $K(\mathbf{y}_i, \mathbf{y}_j; \sigma) = \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{2\sigma^2}\right)$ with bandwidth parameter σ. This kernel has the advantage to be more localized and adaptable using σ.

σ's Choice: The perplexity impacts the distances between items within an embedding. Consequently, σ is adapted by looking at the effective distances by:

$$\hat{\sigma}(Y, k) = \text{Median}\left[\|\mathbf{y} - \mathbf{y}_{k\text{-NN}}\|\right] \tag{5.3}$$

where $\mathbf{y}_{k\text{-NN}}$ is the k-est nearest neighbors of point $\mathbf{y}$, with $k = \lfloor perp \rceil$ and $\|\mathbf{y}\| = \sqrt{\sum_{i=1}^{d} y_i^2}$ is the Euclidian norm. Using the median rather than the mean limits the outliers' contribution which would enlarge σ.

5.2.3 Building the Samples Embedding

The t-SNE algorithm requires as input a distance matrix. Therefore, the distances between samples are obtained by measuring the divergence between the samples' probability vector obtained using Eq. (5.2). We use the Jeffrey-Kullback-Leibler divergence to measure items proximity, with the formulation:

$$KL^J\left(\mathbf{p}(s_a) \| \mathbf{p}(s_b)\right) = \frac{1}{2}\left(KL(\mathbf{p}(s_a) \| \mathbf{p}(s_b)) + KL(\mathbf{p}(s_b) \| \mathbf{p}(s_a))\right) \tag{5.4}$$

with $\mathbf{p}(s_a)$ and $\mathbf{p}(s_b)$ the vectors of sample s_a and s_b respectively. We use this divergence instead of the traditional KL because of the symmetry of KL^J. The matrix $D^S = \left[KL^J\left(\mathbf{p}(s_a) \| \mathbf{p}(s_b)\right)\right]_{s_a, s_b \in \mathcal{S}}$ allows to obtain a new sample embedding $Y^{\mathcal{S}} \leftarrow t\text{-SNE}(D^{\mathcal{S}})$ used in the next iteration inverting samples and features' role.

5.2.4 Embedding Procedure Summary & Parameter Choices

Algorithm 1: Co-Embedding procedure

Input: $(perp^{\mathcal{R}}, perp^{\mathcal{T}})$: Perplexities; k: Number of iterations.

Data: $M = \{\delta(r,t)\}_{r\in\mathcal{R},t\in\mathcal{T}}$: the resources-tags matrix.

Output: $Y^{\mathcal{R}}, Y^{\mathcal{T}}$: Resources and tags respective embedding

$\left(Y^{\mathcal{R}}, Y^{\mathcal{T}}\right) \leftarrow \text{Init}(M)$ `// Initialization`

$(\mathcal{S}, \mathcal{F}) \leftarrow (\mathcal{R}, \mathcal{T})$ `// Resources start with the samples' role`

for $t = 1$ **to** $2k$ **do**

- $\sigma^{\mathcal{F}} \leftarrow f(Y^{\mathcal{F}}, perp^{\mathcal{F}})$ `// Using Eq.` (5.3)
- $P^{\mathcal{S}} = \{\mathbf{p}(s)\}_{s=1:|\mathcal{S}|} \leftarrow g(M, Y^{\mathcal{F}}, \sigma^{\mathcal{F}})$ `// Using Eq.` (5.2)
- $D^{\mathcal{S}} \leftarrow [KL^{J}(\mathbf{p}_a \| \mathbf{p}_b)]_{\mathbf{p}_a,\mathbf{p}_b\in P^{\mathcal{S}}}$ `// Using Eq.` (5.4)
- $Y^{\mathcal{S}} \leftarrow t\text{-SNE}(D^{\mathcal{S}}; perp^{\mathcal{S}})$
- `/* Exchange roles */`
- $(\mathcal{S}, \mathcal{F}) \leftarrow (\mathcal{F}, \mathcal{S})$
- $M \leftarrow M^T$

$(Y^{\mathcal{R}}, Y^{\mathcal{T}}) \leftarrow (Y^{\mathcal{S}}, Y^{\mathcal{F}})$

Initialization: At the start, no initial embedding exists yet, which prevents measuring density. We initialize the first embedding with the d first eigen-vectors of M obtained by SVD. Compared to a random initialization, it fastens the convergence process by starting from an organized state. In pseudocode (1), resources start the role of samples. This is an arbitrary choice and has almost no impact on the final result.

Iteration At each step, the samples' probability density is estimated over the feature's space. Then, samples' divergence matrix is obtained using these probabilities. The embedding step finish by embedding using the t-SNE algorithm to obtain a new sample embedding $Y^{\mathcal{S}}(t+1)$ using the divergence matrix as input. After building the embedding, features and samples exchange their respective roles.

Ending Criterion: The pseudocode (1) iterates k times on each type of item, an arbitrary value around a dozen steps, as the algorithm does not minimize a particular criterion. Nonetheless, the process can be monitored in terms of *neighborhood stability*, looking at if the neighborhood around a point is unchanged over successive embedding. By denoting $\mathcal{N}_n(Y, i)$ the set of the n nearest neighbors of i in Y, we compare its neighborhood in $Y(t)$ and $Y(t+1)$ using the Jaccard similarity by $\frac{|\mathcal{N}_n(Y(t),i)\cap\mathcal{N}_n(Y(t+1),i)|}{|\mathcal{N}_n(Y(t),i)\cup\mathcal{N}_n(Y(t+1),i)|}$.

Algorithm Complexity: For n samples and m features, the probability estimation requires $\mathcal{O}(nm^2)$ operations. The divergence measurement requires $\mathcal{O}(n^2m)$ operation to compute n^2 pairs with vectors of size m. Then, the t-SNE embedding complexity is in $\mathcal{O}(kn^2)$ for k update steps, typically between 100 and 1000, depending on the dataset size and convergence speed. In total, a step described in pseudocode 1 requires $\mathcal{O}\left(nm(n+m) + kn^2\right)$ operations.

5.2.5 Clustering Embeddings

The proposed algorithm leads to clusters' apparition when community structures exist in a dataset. Groups are extracted using the Mean-Shift (MS) clustering algorithm [29]. This algorithm follows an iterative process, moving items towards their *mode*'s location, which are positions with maximal probability density. The image $\bar{\mathbf{y}}$ of point $\mathbf{y}$ is moved toward its mode following the equation:

$$\bar{\mathbf{y}} \leftarrow \frac{\sum_{j=1}^{n} K(\bar{\mathbf{y}}, \mathbf{y}_j)\mathbf{y}_j}{\sum_{j=1}^{n} K(\bar{\mathbf{y}}, \mathbf{y}_j)} \tag{5.5}$$

starting with $\bar{\mathbf{y}} = \mathbf{y}$ and using the Gaussian kernel $K(.)$ defined previously, using the bandwidth parameter σ estimated using Eq. (5.3). After several iteration steps, all items may have converged in some specific locations. A cluster is obtained by gathering all items within a radius ϵ. Using MS, tags clusters $\mathcal{C}^T$ and resources clusters $\mathcal{C}^R$ are extracted from their respective last embedding Y^T and Y^R.

Two-Ways Clustering: Mean-Shift is a *one-way* clustering algorithm as samples and features are clustered independently. Nevertheless, we consider the full process as a *two-way* co-clustering algorithm as the embeddings are linked together,. The MS algorithm has the advantage of being parameter-free, as the kernel bandwidth σ is adapted to the embedding. The two main advantages of MS are its ability to automatically discover the number of clusters and the uniqueness of the partitioning obtained.

Co-Clusters Relationships: A sample is represented by a mixture of features using Eq. (5.2). Two samples s_a and s_b are embedded close to each other area if they have a similar mixture. The features of a sample are not necessarily all located in the same area. Assuming the features of s_a are located in k distinct areas, s_b is similar to s_a only if its features are also located in these k areas. Therefore, we would obtain a co-cluster connected to k features clusters. If s_b has no feature in one location or has a feature in another location out of these k areas, its mixture will be very dissimilar. KL divergence highly penalizes couples of items where one has a mass where the other has none. If s_b has its features located in all these k locations with many features in some of them, an asymmetry can appear due to the excess and deficit of mass in the different areas. Depending on the strength of the asymmetry, s_a and s_b can be located either in the same cluster or in two distinct clusters. A cluster can be characterized by the features clusters it is connected to and the connection's strength.

5.3 Evaluation Methods

5.3.1 Datasets

We propose to evaluate our co-embedding approach over various datasets corresponding to tags associated with resources, as tags allow us to evaluate a group content subjectively. We tried to select for each resource type two datasets to visualize the impact of different data

collection processes. Some of the datasets are folksonomies, tagged by non-expert people with uncontrolled vocabulary, while the others are tagged by experts using a specific vocabulary.

Table 6.1 summarizes the different datasets' characteristics, such as the number of unique resources $|\mathcal{R}|$ and tags $|\mathcal{T}|$, the average number of tags per resources $\mathbb{E}(|r|)$ and the corresponding standard deviation $\sigma(|r|)$, and if the dataset is a folksonomy (*Folks ?*).

Table 5.1: Summary of the (unprocessed) datasets' characteristics

Dataset	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	$\mathbb{E}(\lvert r\rvert)$	$\sigma(\lvert r\rvert)$	Folks ?	Media
BibTex	813,548	255,496	3.3	4.2	yes	bibliography
DBLP	4,894,081	132,337	10.2	1.7	no	bibliography
Corel5K	5,000	347	3.5	0.6	no	image
Flickr	946,113	345,897	11.5	9.6	yes	image
MovieLens	10,381	1,127	44.3	27.7	no	movie
IMDB	120,919	1,001	19.4	12.0	yes	movie
Delicious	16,105	501	18.3	39.5	yes	URL
BibUrl	618,245	156,497	3.4	2.7	yes	URL
Last.fm	445,821	138,402	3.5	3.5	yes	music
NG20	19,300	1,007	32.1	32.4	no	netnews
OHSUMED	13.929	1,002	39.7	17.1	no	abstract

For bibliographic tags, the *BibTex* part of *Bibsonomy* [93] and its non-folksonomy counterpart *DBLP* [27] were selected. For images, we used the *Flickr* folksonomy dataset (MIRFLICKR 25) and the *Corel5K* [94]. The *IMDB* [95] and the *MovieLens20M tag genome* [96] were used for movies. *IMDB* is classified as a folksonomy as words are extracted from movies' reviews, while *MovieLens*'s tags come from a controlled list. While $|\mathcal{T}|$ is similar for both, the vocabulary used are different. Url datasets *Delicious* [98] and the *URL* part of the *Bibsonomy* database [93] (denoted *BibUrl*) were selected. We used the Million Song Dataset from *Last.fm* [49] for music songs but did not find another equivalent dataset. We used textual datasets for comparison. The *NewsGroup20* [99] (NG20) corresponds to news articles covering 20 different topics, and the *OHSUMED* [100] to abstracts from MedLine papers. The *Corel5k*, *Delicious*, *IMDB*, *NewsGroup20* and *OHSUMED* datasets were gathered on https://cometa.com.

The detailed pre-processing steps are detailed in 5.6. The largest datasets are sampled and filtered before use as they do not fit in memory. The sampling and filtering steps are also detailed in the local appendix.

5.3.2 Cluster Retrieval Tasks

We propose to evaluate the co-clusters obtained using our method over a cluster-retrieval task, where the goal is to find the cluster where the item's location is. Additionally, we evaluate the ability of the features clusters to serve as a unit of description. The sample's binary vector is replaced by the proportion of features it has in each cluster, providing a

compact representation. For a sample s, its description vector is:

$$\mathbf{q}(s) = \left[\frac{\sum_{f \in C^F} \delta(s, f)}{\sum_{f \in F} \delta(s, f)} \right]_{C^F \in \mathcal{C}^F} \tag{5.6}$$

as well as for a cluster $C \in \mathcal{C}^S$ by:

$$\mathbf{q}(C) = \left[\frac{\sum_{s \in C, f \in C^F} \delta(s, f)}{\sum_{s \in C, f \in F} \delta(s, f)} \right]_{C^F \in \mathcal{C}^F} \tag{5.7}$$

The compressed vectors are denoted by $\mathbf{q}$ to avoid confusion with the vectors $\mathbf{p}$ defined in Eq. (5.2). We use the *binary* features and not the *diffused* ones here as the goal is to evaluate the partitioning obtained, and not the embedding quality. Clusters $C \in \mathcal{C}^S$ are sorted by increasing KL divergence $D(s, C) = KL(\mathbf{q}(s) \| \mathbf{q}(C))$. The retrieval accuracy of this task is measured using the *Mean Retrieval Rank* (MRR), which quickly drops with miss-prediction.

The retrieval task allows us to evaluate the partitioning relevance. This is assessed by the capabilities of sample clusters to gather items with similar connections, and by the ability of feature clusters to serve as a description unit. Cluster partitioning can help to speed up the retrieval process. A user query is often short and inaccurate, which makes impossible an exact search. The query must be compared to all n items of the database to select the most relevant item. By partitioning into k clusters, the search requires k operations to find the best cluster (assuming the correct cluster is well identified), and $\frac{n}{k}$ others to find the correct element within the cluster. Therefore, the partitioning can help to reduce the search complexity from $\mathcal{O}(n)$ to $\mathcal{O}(k + \frac{n}{k})$, which could be beneficial for very large or distributed databases.

5.3.3 Comparative Algorithm

We compare our algorithm to the spectral co-clustering algorithm presented in [63]. The algorithm uses the VSM representation that SVD next decomposes. Then, the singular vectors are clustered using k-Means. The algorithm is adapted by grouping the two types of nodes separately. As this algorithm has to enter the number of clusters to be searched, we use the number of clusters found using our approach. k-Means is a non-deterministic algorithm leading to different results depending on the initialization seed. The algorithm is run 10 times, and the best result is kept for fairness. We choose this algorithm as it has a strong theoretical basis but does not consider dimension relatedness.

5.3.4 Evaluation Metrics

5.3.4.1 Cluster Quality

Not all partitioning with k clusters are equivalent. The proposed task is very easy if a single cluster gathers all the items, and the other clusters are made of very small groups. In contrast, the retrieval task using partitioning with clusters of equivalent size is much harder, as a

random choice based on the mass would lead to very poor scores. We define the partitioning entropy as the entropy relative to the clusters' size:

$$H(\mathcal{C}) = -\sum_{C \in \mathcal{C}} Pr(C) \log_2 \left(Pr(C)\right) \tag{5.8}$$

where $Pr(C) = \frac{|C|}{\sum_{C_i \in \mathcal{C}} |C_i|}$, with values in $[0, \log_2 |\mathcal{C}|]$.

The value in terms of entropy is non-comparable to the number of clusters as it represents the number of bits necessary to encode the information. The value is put to the power of 2 to transform it into a meaningful value. The effective number of cluster $k(\mathcal{C}) = 2^{H(\mathcal{P})} \in [1, |\mathcal{C}|]$, where a value of 1 is synonymous with a large cluster, while a value close to $|\mathcal{C}|$ indicates that the clusters have a similar size. This value better measures the retrieval difficulty, as clusters' size is considered.

5.3.4.2 Items Representativeness

The items in the identified clusters can be ranked by *relevance* or *representativeness*. A sample is relevant if its distribution on the feature space is close to the mean distribution of the cluster to which it belongs to.

We propose to score items based on their KL-divergence from their respective cluster. The KL-divergence could be expressed as $KL(A\|B) = H^*(A, B) - H(A)$, where $H^*(.)$ is the cross-entropy that represents the extra cost of coding A with the optimal code of B. For a cluster $C \in \mathcal{C}^{\mathcal{S}}$, we define the representativeness of sample $s \in C$ relatively to C as:

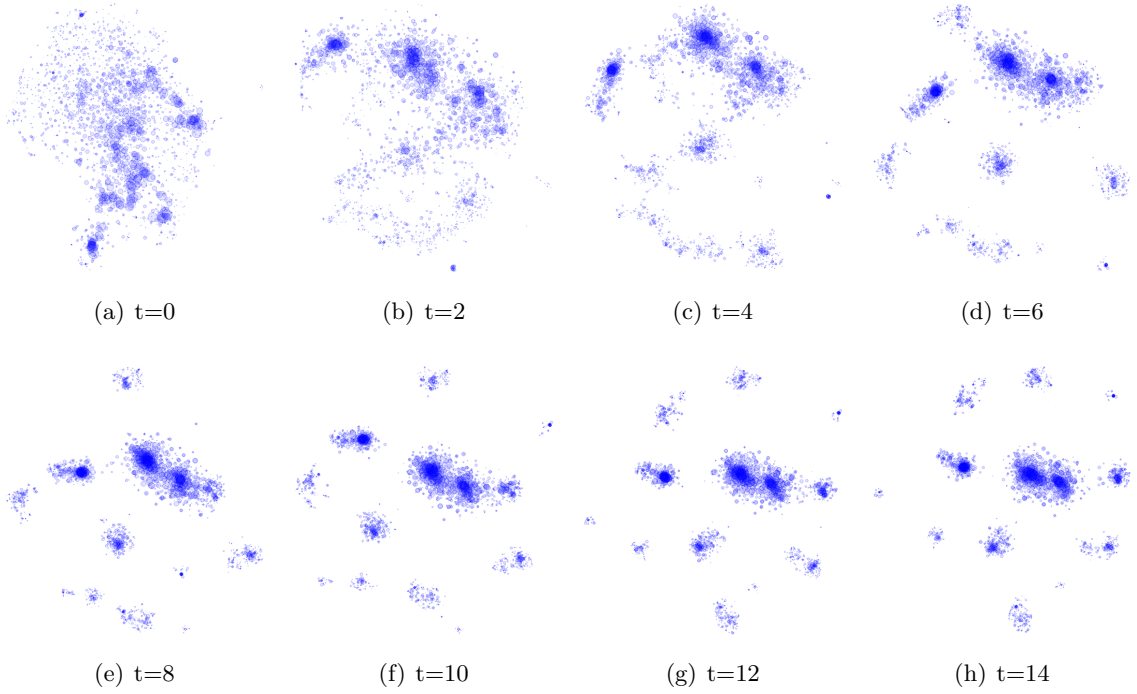
$$\text{repr}(s, C) = \max\left(0, 1 - \frac{KL(\mathbf{q}(s)\|\mathbf{q}(C))}{H(\mathbf{q}(s))}\right) \tag{5.9}$$

A score $\text{repr}(s, C) \in [0, 1]$ is obtained, where 1 represents the maximal relevance.

This normalization enables us to consider samples' frequency. A low divergence between the item and the cluster is synonymous with high similarity. However, underfrequent items are likely to have a lower divergence than frequent items. By normalizing the divergence with the entropy, items are fairly compared to the cluster average.

5.4 Results

5.4.1 Evolution over Time



(a) t=0 (b) t=2 (c) t=4 (d) t=6

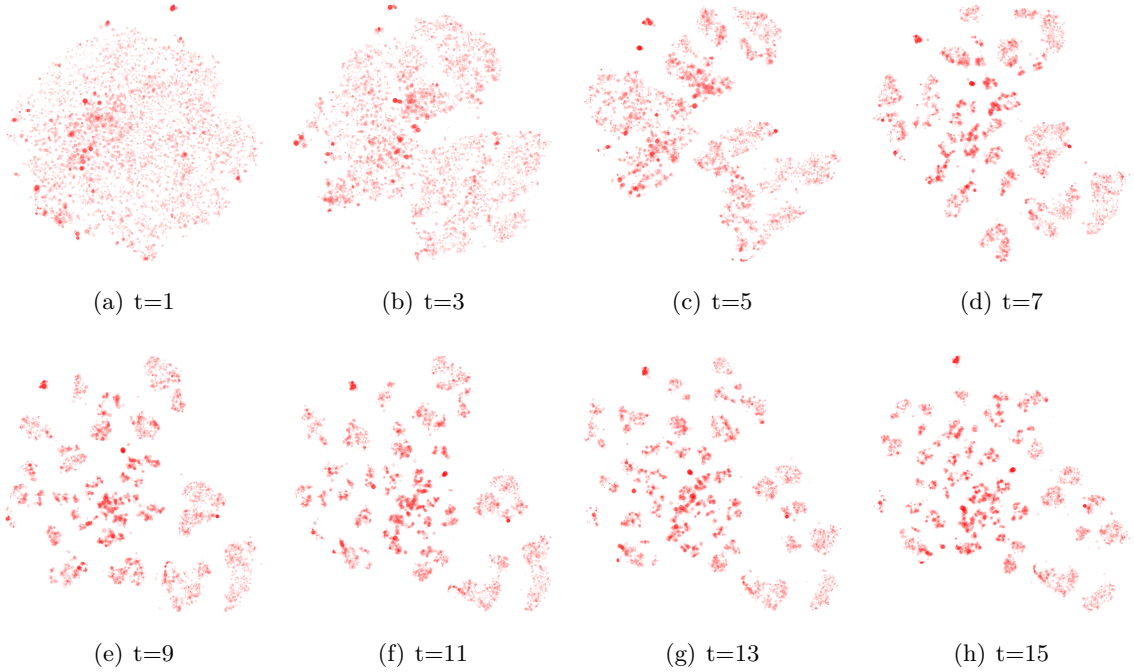
(e) t=8 (f) t=10 (g) t=12 (h) t=14

Figure 5.1: Evolution of the *keywords* co-embedding for the *Flickr* dataset.

Fig. 5.1 and Fig. 5.2 represent the embedding chronology for keywords and resources respectively, for randomly selected items of the *Flickr* dataset. The subset is described in Table 5.2 and the resulting co-embedding is reused in the next experiments.

t-SNE is initialized with the previous items' location, allowing to visualize items displacement over time better. Therefore, clusters' positions are approximately preserved over the successive steps. The first keyword embedding ($t = 0$) is obtained by exploiting the document eigenvectors obtained by SVD decomposition, while the first document embedding ($t = 1$) is obtained exploiting this first keyword embedding.

Major structures emerge from the initial mass of items during the first steps (until $t = 6 \sim 7$). During the last steps, clusters' shapes are refined. The keyword clusters' locations are stable over time, but clusters tend to densify from step $t = 6$ to step $t = 14$, leading to well-separated clusters. Keyword clusters' move from their previous location and some split into several pieces between $t = 7$ and $t = 15$. At the end of the process, both embeddings show clusterable structure but differ in their spatial organization. The difference will be detailed in the next paragraphs.



(a) t=1 (b) t=3 (c) t=5 (d) t=7

(e) t=9 (f) t=11 (g) t=13 (h) t=15

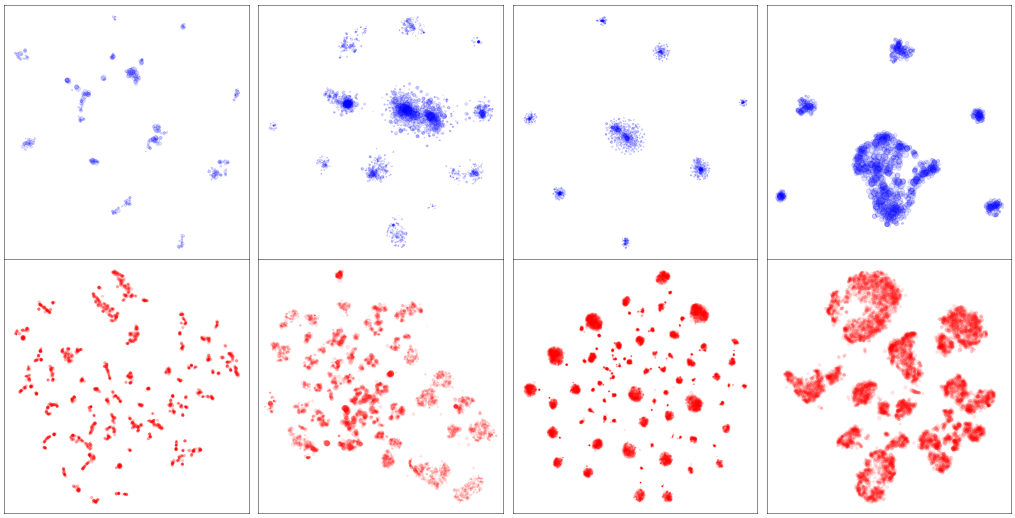
Figure 5.2: Evolution of the *documents* co-embedding for the *Flickr* dataset.

5.4.2 Visual Results

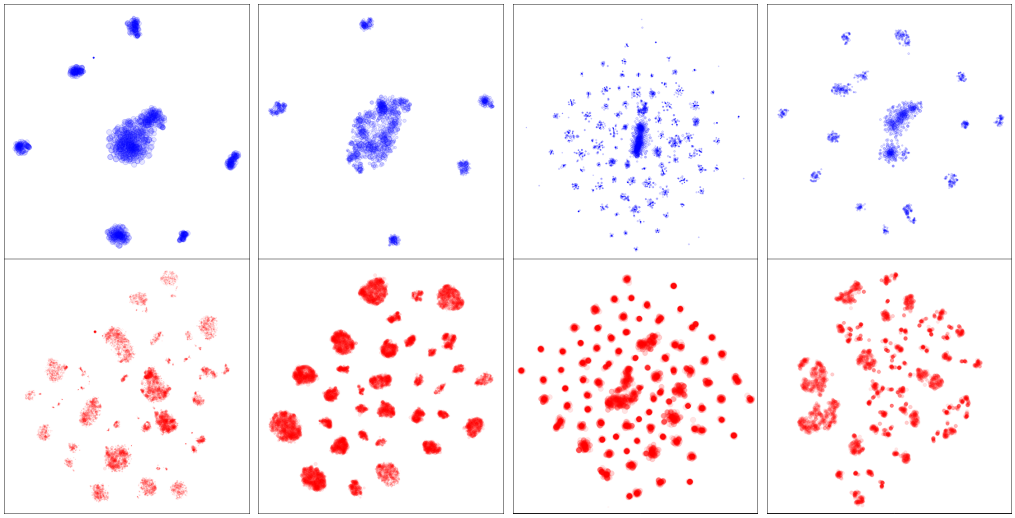
Fig. 5.3 shows different co-embedding results for the different datasets, where samples' characteristics are presented in Table 5.2. The majority presents disjoint clusters for both tags and resources. The tag embeddings differ from the resource embeddings as they all show a large central cluster. These clusters correspond to unspecific vocabulary, which occurs in all resources' clusters. There is no such a central cluster for resources' clusters, unless for the unfiltered *DBLP* subset and *Delicious*.

Datasets are filtered to fit in memory by discarding rare items as they are the less accurate and would lead to a *normalization bias* (5.5.2). We illustrate the difference between the *unfiltered* (u) and *filtered* (f) over a *DBLP* subset related to the *payment*. The filtering leads to removing many infrequent tags, drastically reducing $|\mathcal{T}|$ and $|\mathcal{C}_T|$. $|\mathcal{R}|$ is unaffected by the filtering, but $|\mathcal{C}_R|$ is reduced by 1/7, leading to denser clusters. The reduction of $|\mathcal{C}_R|$ results from removing these infrequent tags that isolated some resources into small clusters.

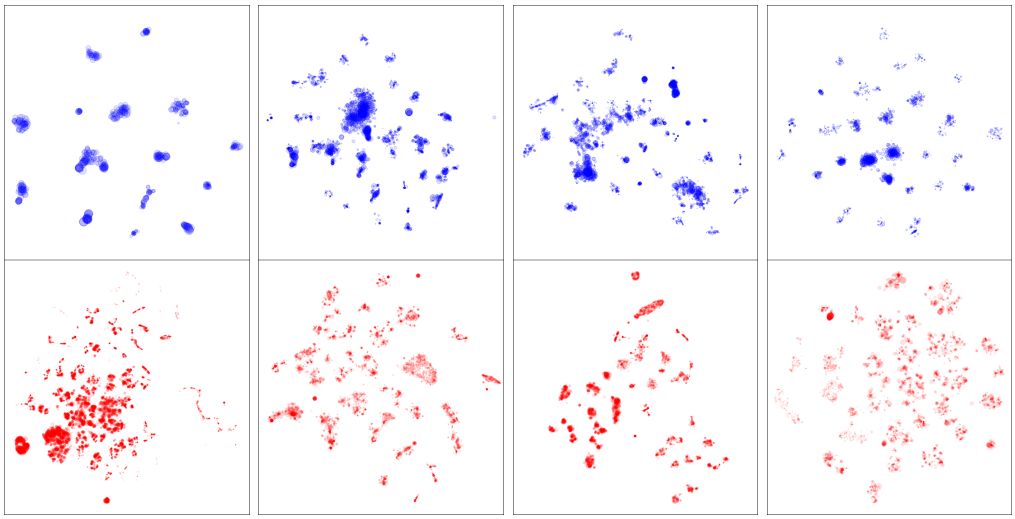
Some datasets have a different clusters' shape and spatial organization. At the same time, most tag clusters are dense; the *Corel5K* has very thin clusters. This could be explained by the very low $\mathbb{E}(|r|)$, and the small tag collection, leading to very weak connectivity. The *Bibsonomy* datasets have some of their tag clusters connected to each other. These clusters gather similar vocabulary but with different formatting. By applying text processing methods (steaming and case normalization), these clusters would merge in one.



(a) Corel5K (b) Flickr (c) IMDB (d) MovieLens



(e) NG20 (f) OHSUMED (g) DBLP (u) (h) DBLP (f)



(i) Delicious (j) BibUrl (k) BibTex (l) Last.fm

Figure 5.3: Co-embedding results. The blue items on the top correspond to tag embeddings while the red ones to resource embeddings. Opacity depends on area crowdedness and items' frequency (light color for rare items).

5.4.3 Cluster Balance

The characteristics of the selected subsets used to build the co-embedding presented in Fig. 5.3 are detailed in Table 5.2. The subsets dimensions $|\mathcal{R}|$ and $|\mathcal{R}|$ are limited to thousands of items to fit in memory. The table gathers the number of clusters ($|\mathcal{C}_R|$ an $|\mathcal{C}_T|$) obtained by clustering the co-embeddings (CE) with Mean-Shift, and the effective number of clusters $k(\mathcal{C}^{CE})$ measured with Eq. (5.8). The MS clustering results are compared to the raw binary matrix's spectral clustering (SC), searching for the same number of clusters $|\mathcal{C}|$.

Table 5.2: Sample size, number of clusters, and effective number of clusters using our co-embedding approach (CE) and spectral clustering (SC).

Dataset	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{C}_R\rvert$	$k\left(\mathcal{C}_R^{CE}\right)$	$k\left(\mathcal{C}_R^{SC}\right)$	$\lvert\mathcal{C}_T\rvert$	$k\left(\mathcal{C}_T^{CE}\right)$	$k\left(\mathcal{C}_T^{SC}\right)$
BibTex	7,643	4,335	54	**35.4**	20.7	45	**35.9**	15.9
DBLP (u)	5,028	5,624	114	**71.8**	48.1	102	**76.9**	41.95
DBLP (f)	4,948	1,443	81	40.2	**61.4**	16	13.2	**13.5**
Corel5K	5,000	364	88	**62.5**	60.9	14	**12.3**	7.1
Flickr	7,185	2,427	70	**34.2**	26.7	16	**13.0**	1.9
MovieLens	10,159	1,127	21	9.5	**18.9**	10	7.8	7.8
IMDB	16,000	1,001	125	65.3	**90.1**	8	**6.4**	5.7
Delicious	16,105	501	118	**76.2**	41.1	9	**8.4**	4.2
BibUrl	7,886	3,648	45	**25.3**	9.8	38	**27.6**	9.0
Last.fm	9,897	2,516	66	**32.9**	15.5	32	**29.5**	5.8
NG20	13,125	1,007	59	27.8	**48.5**	9	**6.8**	4.2
OHSUMED	13,929	1,002	34	26.7	**31.9**	7	5.1	**6.4**

As a general remark, tag clusters are less numerous than resource clusters as observed in the previous figure. This could be explained by the fact that $|\mathcal{R}| > |\mathcal{T}|$ for all subsets. However, there is no explicit relationship between sample size and clusters number, as *DBLP* has more resources clusters than most other datasets while having the smallest $|\mathcal{R}|$. Thus, the number of clusters found is likely to depend on the sample intrinsic characteristics.

$k(\mathcal{C}_T)$ is almost always larger for CE than for SC, while half of the time for $k(\mathcal{C}_R)$. $k(\mathcal{C}^{CE})$ and $k(\mathcal{C}^{SC})$ have the same order of magnitude, unless for a few examples like *Flickr* and *Last.fm* with a very low $k(\mathcal{C}_T^{SC})$ compared to $|\mathcal{C}_T|$, and *BibUrl* where both $k(\mathcal{C}_R^{SC})$ and $k(\mathcal{C}_T^{SC})$ are much lower than for CE.

One reason that might explain the difficulty to identify balanced clusters using the raw binary matrix with SC is the power-law distribution of items. This distribution leads to a very sparse matrix where groups are difficult to identify. Resource occurrences follow a power-law if some resources are more popular therefore more tagged than others, but the phenomenon is more frequent for tags because of the language organization. This might explain why SC gives lower results than CE more frequently over tags than over resources.

5.4.4 Representative Keywords

Table 5.3: Top tags for the largest cluster (denoted *Main group*) and four other clusters for each dataset. Acronyms: CS: Computer Science, CVA: Common Value Auction, HCI: Human Computer Interaction, NN: Neural Networks, Mech.: Mechanism

Dataset	Main group	Group 2	Group 3	Group 4	Group 5
BibTex	Middle	Recording	Cell	Inflation	Joint
	of	Videotape	Protein	Geld	Hip
	Humans	Video	Models	Kapitalmarkt	Dislocation
	Cerebral	Equilibrium	Membrane	EU-staaten	Ostonomy
DBLP (f)	CS	CVA	Use case	3-D Secure	Multimedia
	Payment	Revenue	USable	Credential	Analytics
	DB transaction	Op. research	Web service	MULTOS	HCI
	The Internet	Mech. design	Web app.	OpenPGP card	Artif. NN
Corel5K	water	canal	tundra	train	deer
	sky	sailboats	polar	plane	elk
	tree	lake	bear	railroad	white-tailed
	hills	dock	storm	locomotive	antlers
Flickr	light	pontrouge	mare	meiji	spiegelung
	blue	pontneuf	nuvole	era	landskap
	nikon	collette	tramonto	period	flickrolf
	city	impressionniste	acque	prints	reise
MovieLens	talky	splatter	scifi	us history	crime
	cinematography	gory	space	war	thriller
	criterion	horror	sci-fi	wartime	mystery
	melancholic	demons	sci fi	ethnic conflict	murder
IMDB	life	films	american	father	game
	year	director	film	finds	team
	time	video	movie	wife	star
	day	documentary	war	friends	battle
BibUrl	web	Post-abortion	assistive	stanford	kittens
	blog	Pro-life	impaired	ml	cats
	reference	Lebensrecht	visually	bioinformatics	memetics
	blogs	Human Rights	cane	model	cute
Delicious	throat	das	breath	apache	him
	knees	ein	loud	article	looked
	leaning	den	mouth	rails	told
	rodney	der	neck	ruby	couldn
Last.fm	pop	Hamburger Schule	sixties	tech-house	country pop
	indie	The Killers	1960s	seep techno	switzeland
	rock	lofi	1960's	melodic trance	discover
	alternative	elliott smith	70's	dream trance	daytrotter
NG20	time	work	religious	windows	law
	don	problem	jewish	software	country
	make	mail	religion	file	government
	article	system	god	ms	rights
OHSUMED	patients	compared	artery	antibody	surgery
	disease	greater	left	antigen	surgical
	findings	4	cardiac	immune	complication
	clinical	group	anterior	antibody	procedure

The top 4 words of each cluster were extracted using Eq. 5.9 from the clusters identified previously using CE. As a result, the keywords of the largest clusters (denoted *Main group*) and four other selected clusters are listed in Table 5.3.

Main Group Description The main cluster of each subset is mostly made of general words occurring in many different contexts that the dataset cover. In some datasets, it allows to define the global scope (music genres are identified for *Last.fm*, medical topics for *OHSUMED*, or web components for *BibUrl*), but for the majority the media type (music, images, etc.) or the topics covered are difficult to identify.

Cluster Identity Each cluster has a particular identity. For example, for *OHSUMED*, one group gathers the vocabulary used in studies (where cohorts are compared), and the others concern different disciplines (related to heart, infectiology, and surgery). For *Corel5K*, each group corresponds to a particular picture (seaside, tundra, locomotion engines, and animals).

A folksonomy includes vocabulary from users that do not necessarily share the same language. *Flickr* is the most representative dataset, with French, Italian, English and German words each gathered in a specific cluster. This particularity is visible on the other folksonomies *BibTex*, *BibUrl* and *Delicious* with English and German clusters.

In a given dataset, words groups can be of various kinds, out of language consideration. For example, the *Last.fm* dataset has five clusters with distinct words types and ideas. The main group is about general music type while the 4th gathers sub-music types related to electronic dance music, the 2^{nd} gathers band names, the 3^{rd} epochs, and the last road-trip related ideas.

Data Collection Specificity While several datasets have in common the type of tagged resources; the clusters obtained differ in the ideas expressed. This could be explained in a first place by the small overlap between the dataset's vocabulary, but the ideas expressed differ from one dataset to another.

Corel5K gathers timeless tags each representing a specific object or place, while *Flickr* gathers tags related to art requiring culture to be understood. The difference can be explained by the different goals expected, where *Corel5K* is designed for Object Recognition, while information retrieval for Flickr.

The largest difference concerns *IMDB* and *MovieLens* that both correspond to movies' descriptions with approximately the same number of tags. Movie styles are easily identified on the *MovieLens* dataset (horror, science-fiction, historical and police movies), while *IMDB*' clusters group elements that occur in the same context together. Each dataset has a narrowed list of terms, corresponding to the most relevant term for *IMDB*, and general movies attribute [96] for the *MovieLens* tag genome, explaining the difference of expressed ideas between the two datasets.

5.4.5 Cluster Retrieval

We tested the cluster retrieval tasks using the clusters obtained previously. In addition, we tried the task in both directions: searching the resource's cluster, and searching the tag's cluster. Table 5.4 summarizes the retrieval scores for this task.

Table 5.4: Mean Retrieval Rank for resources clusters $\mathcal{C}_R$ and tag clusters $\mathcal{C}_T$ using our approach (CE) and the comparative algorithm (SC).

Dataset	$MRR(\mathcal{C}_R^{CE})$	$MRR(\mathcal{C}_R^{SC})$	$MRR(\mathcal{C}_T^{CE})$	$MRR(\mathcal{C}_T^{SC})$
Bibtex	**94.1** %	89.1 %	**92.5** %	89.6 %
DBLP (u)	**86.7** %	70.3 %	**97.8** %	73.6 %
DBLP (f)	**91.5** %	66.2 %	**99.6** %	88.7 %
Corel5K	**69.2** %	48.5 %	**99.3** %	90.46 %
Flickr	**78.5** %	20.9 %	**96.2** %	86.74 %
IMDB	**96.4** %	34.7 %	**100.0** %	92.9 %
MovieLens	62.8 %	**79.2** %	86.9 %	**93.6** %
BibUrl	87.1 %	**89.0** %	90.4 %	**91.7** %
Delicious	**74.9** %	53.7 %	**98.1** %	97.5 %
Last.fm	**83.8** %	70.0 %	**97.9** %	85.2 %
NG20	**88.7** %	49.9 %	**98.2** %	96.1 %
OHSUMED	**91.4** %	50.6 %	**97.9** %	91.5 %

Tags clusters have higher retrieval scores than resources clusters with both approaches, ours leading to better MRRs in most cases. The good MRRs are explained by the low number of tag clusters, making the task easier with a smaller choice. The other point explaining these MRRs is many resource clusters, allowing a more detailed description vector, leading to a more accurate retrieval. The argumentation is reversed for resource cluster retrieval, leading to lower MRR scores. The *Flickr* dataset mentioned for its low $k(C_T^{SC})$ (1.9) illustrates this argument with the lowest retrieval score $MRR(\mathcal{C}_R^{SC}) = 20.9\%$. A low number of tags clusters inevitably leads to a low retrieval rate of the resource clusters. Still, a large number doesn't guarantee the opposite outcome, illustrated by the *DBLP* dataset, with an acceptable retrieval score using our approach while having the largest number of tag clusters.

The gap between approaches for resource retrieval is larger, with our approach having MRR around $80 \sim 90\%$, the spectral clustering has MRR around $50 \sim 70\%$. These results make our approach preferable for this particular task, with the additional advantage of providing an intermediate visualization result.

5.5 Discussion

5.5.1 Parameters Choice

Perplexity As said previously, the perplexity governs the embedding shape by controlling the number of items t-SNE would look at for embedding data. The values we used must be relatively small because we are interested in local structures to learn similarities. These values are adjusted as a function of the sample size n the following way:

$$perp = \begin{cases} 15 & \text{if } n < 1,000 \\ 30 & \text{if } 1,000 \leq n \leq 8,000 \\ 60 & \text{else} \end{cases}$$

This decision rule is very simple and allows a simple parameter selection. Nevertheless, the verification of the produced output is suggested. A too large value would connect the different clusters together, while a smaller value would lead to more clusters, penalizing the other embedding.

Relationships between σ and $perp$: The estimation of σ using Eq. (5.3) can be explained with two remarks. First, a larger perplexity leads to smaller distances between nearest neighbors (NN), making the choice of σ non-trivial. The direct adaptation to the embedding allows adapting to the perplexity and the dataset characteristics. Second, the perplexity governs the number of NN that t-SNE considers around each item. If $k < perp$, the kernel would underexploit the items' positioning, as only a small fraction of the NN would be considered. On the opposite case where $k > perp$, the kernel would be too large, and would put weights on items that were not considered by t-SNE. Additionally, a larger σ would lead to a flatter kernel, underexploiting items well positioned. Therefore, the best option is to select $k = \lfloor perp \rceil$, simplifying σ's choice by adapting it indirectly to the current perplexity.

5.5.2 Normalization

Normalization Bias: In Eq. (5.2), the contribution of an item is normalized by its frequency, limiting the contribution of highly popular items. In contrast, items with very few links and low frequency have their contribution enhanced. Their weight is very large for very low-frequency items, leading to *group artefacts*, where the infrequent item is strong enough to gather co-occurring items in a single cluster. To overcome this issue, using a different normalization scheme or filtering the dataset could mitigate the apparition of artefacts.

Weighted Case: Our approach exclusively considers the case where the frequency information is unavailable. Nonetheless, Eq. (5.2) can be adapted to consider items' frequency, by replacing $\sum_{f_i \in \mathcal{F}} \frac{\delta(s,f_i)}{|f_i|}$ by $\sum_{f_i \in \mathcal{F}} \frac{TF(s,f_i)}{\sum_{s_j \in \mathcal{S}} TF(s_j,f_i)}$ where $TF(s_j, f_i) = \frac{C(s_j,f_i)}{\sum_{f' \in \mathcal{F}} C(s_j,f')}$ is the frequency of feature f_i in sample s_j and $C(s_j, f_i)$ the number of occurrences. If we have words and documents, the normalization of the word count within a document is intuitive,

allowing to obtain a document vector summing to 1. However, describing a word using the documents that contain it to obtain a word vector is non-trivial as different possibilities exist. One can normalize the word frequency of each document $\frac{TF(w,r)}{\sum_{r' \in \mathcal{R}} TF(w,r')}$, or one can normalize the word count per document over all the corpus $\frac{C(w,r)}{\sum_{r' \in \mathcal{R}} C(w,r')}$. The first option is more balanced, as it gives credit to all documents. Nevertheless, documents with very few words would penalize the final result because it is considered inaccurate information. In that case, the second option is more suitable, as it favors the contribution of documents of sufficient size.

5.5.3 Scalability to Large Datasets

Each step of our algorithm runs in $\mathcal{O}\left(nm(n+m) + kn^2\right)$, unaffordable for very large datasets. Nevertheless, the scalability problem can be bypassed by identifying clusters over a subset of items. Of course, this assumes that sampling would only affect clusters' size, but would not affect clusters' number. After the cluster identification phase, the unselected items are assigned to the most relevant cluster, classifying them.

Then, the embedding process could be repeated over the items of a consolidated cluster, leading to a new segmentation level. Some features were previously discarded because of their low frequency within the initial subset because of the filtering process. The addition of these new items allows some of these features to reach the critical frequency, therefore to consider them. Even if the items are very similar, the second co-embedding process will likely to lead to new clusters because of the new features that increase diversity.

5.5.4 Adaptation to Temporal Datasets

A dataset is a snapshot of a database at a given moment. A real-world graph is often dynamic, with new nodes and new edges. Rather than looking at the entire network since origin, the focus can be put over a given period to compare with another one.

The proposed approach can be adapted to temporal datasets using a moving window. The first slice is used to construct a primer co-embedding. Next, the embedding of the new window is obtained using the embeddings of the previous slice to learn items' density. This step is possible only if the previous and current windows share some of their elements and new items have already seen features. Otherwise, it would not be possible to evaluate the density. Following this process would reduce the number of co-embedding iterations as an already stable embedding allows estimating items' density. A minor difficulty concerns cluster tracking over the different windows, as new clusters can emerge, disappear, split or fuse together, adding some complexity.

5.6 Conclusion

In this article, we proposed a co-clustering algorithm for bipartite graphs. The algorithm addresses *dimension relatedness* by projecting features into a low dimensional space, comparing samples based on their mixture of features. The embedding process leads to natural cluster formation for a dataset where community structures exist, clustered using the Mean-Shift algorithm. The algorithm is easy to configure with very few parameters to adjust. We tested our algorithm over a cluster retrieval problem. A better retrieval accuracy with more balanced clusters in a large number of cases was shown than when using a spectral co-clustering algorithm that did not consider the relationship between the dimensions. However, the weakest point of our algorithm is scalability. Nevertheless, clusters could be identified over a subset, and unused items assigned to the closer cluster. The approach could be used to optimize recommender systems and retrieval engines by working at the cluster-level rather than the item-level.

As future works, there are multiple directions to explore. As underlined in the discussion, the main limiting factor is the scalability. The first direction to explore is how to sample the dataset to identify the main groups. The co-clustering process can be repeated at the cluster level by assigning unselected items to the closest clusters, creating a hierarchical structure. This recursive idea must be tested to see whereas obtained clusters are meaningful as it would lead to a computationally cheap divisive hierarchical algorithm. Another direction is the usefulness for recommender systems (RecSys). In this article, we tested the retrievability of an item given a query. A second major aspect to consider in a RecSys is the suggestion capability of unseen items to a user. Further works need to be done on this direction to measure the usability of such a clustering to this domain.

Appendix

Sampling & Filtering

Because of the complexity of our proposed method, not all the datasets could be processed as they couldn't fit in the memory (*Bibtex*, *DBLP*, *Flickr*, *IMDB*, and *Last.fm*). Therefore, a first sampling is performed over the documents, trying to keep around 10.000 documents. Then, some documents having too few keywords (less than 5) are removed from the subset. The same filtering is applied to keywords with less than 5 occurrences. The filtering process is repeated until all documents have 5 valid tags and reciprocally.

Dataset Pre-Processing

BibTex & BibUrl The Bibsonomy dataset [93] is composed of two sub-dataset: the former (*BibUrl*) correspond to tagged URLs, while the second (*BibText*)to tagged scientific articles. We use the scientific part, denoted *BibTex*. This dataset is a folksonomy, where users are free to add any keywords to any URL / article. We construct the bipartite graph by preserving

the (document, tag) part, forgetting the user part. Because of the folksonomy origin, the keywords are power-law distributed. We couldn't perform an initial document selection looking at a particular keyword, because the resulting subset after filtering would be too small. Therefore, documents were sampled at random from the whole corpus.

Corel5K The *Corel5K* dataset [94] contains 5 thousand images associated with tags. The dataset has been obtained on the Cometa website (https://cometa.ujaen.es/datasets/). Each image is associated with a set of tags, where no frequency is available. Due to the limited size of this dataset, no sampling nor filtering have been done.

DBLP We used the *DBLP* v. 12 (2020-04-09) citation dataset [27]. The dataset corresponds to scientific articles with information such as title, date, abstract, authors, references, around 10 keywords and other fields. The bipartite graph was build using the associated keywords. Because of the dataset size, a subset of article was extracted. A general keyword was selected (in the experiment: *Payment*), and all documents associated with this keyword gathered. Then, the sampling is performed followed by the filtering step.

Flickr We used the *Flickr* dataset [101] (MIRFLICKR-1M) that contains 1 million of images and the corresponding tags. We removed from the described corpus all images with no associated tags. Then, we applied the sampling and filtering steps.

IMDB The *IMDB* dataset [95] has been gathered on the Cometa website. Each movie is associated with a set of tags as well as a set of classes, representing the movie types. The bipartite graph is constructed using the tags and movies exclusively, and does not use the movie types. For the experiments, the dataset was sampled and filtered.

Last.fm The Million Song Dataset [49] corresponds to music titles associated with rated attributes in $[0, 100]$. The dataset is organized into three part: *train*, *test* and *subset*. We worked only on the *train* part because of the large number of music songs available. As for the MovieLens dataset, we preserved all the attributes with relevance ≥ 80, and discarded the songs with no items. This dataset has been sampled and filtered for the experiments.

MovieLens From the *MovieLens* project [96], we selected the ml-20m dataset and studied the *genome* part. The *genome* is a set of about 1.000 keywords for which the relevance to each movie ($\in [0, 1]$ has been evaluated. For each document, we preserved all the keywords with relevance equal or greater than 0.5. The subset size has been sampled and filtered for the experiments.

NewsGroup20 The NewsGroup20 dataset [99] has been obtained on the Cometa website. For scalability, we only removed the documents with the lowest number of tags. We did not apply the filtering process as the number of tags per document is large, as well as the number of documents including a specific tag. Therefore, the filtering would have no consequence.

OHSUMED The OHSUMED dataset [100] corresponds to medical article abstracts. It has been obtained on the Cometa website without processing.

Part III

Computation under Noisy Condition

CHAPTER 6

Noise-Resilient Ensemble Learning using Evidence Accumulation

Ensemble Learning (EL) methods combine multiple algorithms performing the same task to build a group with superior quality. These systems are well adapted to the distributed setup, where each peer or machine of the network hosts one algorithm and communicate its results to its peers. EL are naturally resilient to the absence of several peers thanks to the ensemble redundancy. However, the network can be corrupted, altering the prediction accuracy of a peer, which has a deleterious effect on the ensemble quality. In this paper, we propose a noise-resilient ensemble classification method, which helps to improve accuracy and correct random errors. The approach is inspired by Evidence Accumulation Clustering (EAC), adapted to classification ensembles. We compared it to the naive voter model over four multi-class datasets. Our model showed a greater resilience, allowing us to recover prediction under a very high noise level. In addition as the method is based on the EAC, our method is highly flexible as it can combines classifiers with different label definitions.

6.1 Introduction

Ensemble Learning [102] (EL) methods combine several algorithms performing the same task to obtain a better-quality group. EL methods play on diverse group aspects: the number of algorithms [103,104], their weighting based on their contribution [105–107], and their selection based on their diversity [108].

EL methods are well adapted to the distributed setup, where several machines host each a single algorithm and send their results to a central node aggregating the results [109–111]. They can be adapted to decentralized Peer-to-Peer (P2P) networks [110], where a dynamic group collaborate to improve its accuracy by electing a leader or by aggregating the group's results. Distributed systems and P2P networks are prone to network failures, where communications are broken between some nodes, or corrupted with noise [112]. In addition, no direct control can be made on P2P nodes, where malicious peers can join at any time, and peers change temporary behavior.

EL methods are resilient to the absence of one or more weak learners thanks to group redundancy [108,113]. However, the corruption of a learner's predictions is equivalent to a negative change of accuracy, which has a deleterious effect on the group quality. Thus, there are two ways to deal with corrupted computers: detecting inaccurate peers to avoid data pollution or resilience to error. The detection can be done using network monitoring methods, or exploiting trust to weigh peers based on their past contributions. However, this approach is not adapted to a dynamic environment such as a P2P network where a peer lifetime is very short and may change temporary behavior. In contrast, being resilient to error is more suitable as all inputs are accepted but more challenging to design as it requires smart correction algorithms.

In this article, we propose a noise-resilient ensemble classification method, correcting errors while improving accuracy. The method uses the Evidence Accumulation Clustering (EAC) approach to rectify class boundaries and correct corrupted labels by performing a local weighted vote. The approach was tested under several noise condition over four datasets and tolerated high noise levels without accuracy degradation.

This paper is structured as follows. The first section presents the related works regarding EL methods and resilience to error. The second section details the proposed ensemble classification method. The datasets and the classifiers' setup are detailed in the experimental section, followed by the results. Finally, the paper ends with a discussion and a conclusion.

6.2 Related Works

Ensemble classification (EClass) methods adopt different strategies to increase accuracy. The simplest one is *bagging* [103] – for *bootstrap aggregating*, learners are trained over randomly sampled subsets. A related approach exploits *random projections* [104], which creates a different view of the data, making it less dependent on the pre-processing step. Rather than sampling items at random, selection can be made on features [114], reducing the computational complexity of each classifier while allowing the ensemble to classify incomplete items with missing features. A classifier often makes very few errors in very dense areas where a single class is represented, because there is no ambiguity. Near a boundary, the classifier is less exact because this area is less dense; therefore difficult to learn correctly. EL methods help mostly to correct these areas. A good example is decision trees with rough decision boundaries, while a *Random Forest* [115] has smooth boundaries with various shapes. The *bagging* approaches exploit the fact that random initialization would lead to diverse classifiers, compensating errors within the group. However, it requires a large number of classifiers and fails to improve when classifiers are correlated.

Boosting [105] solves the correlation problem by weighting classifiers based on the improvement they can lead to. On the other hand, *ensemble pruning* [108] removes correlated classifiers that lead to no improvement, decreasing the overall complexity while preserving the accuracy. The ensemble size is smaller for these methods but is still resilient to the absence of several classifiers during the inference stage [113]. *Boosting* and *ensemble pruning* target the problem of *how to construct* a full ensemble using very few classifiers. As they require a setup phase to evaluate the classifiers, these approaches are not adapted to dynamic environments, such as P2P networks, as peers can join and leave at any time.

Most EClass methods assume that no issue can occur within the ensemble, i.e. all predictions are transmitted without errors. The centralizing node is always available, but some classification peers can be down. In the case of Internet-of-Things networks, like Wireless Sensor Network (WSN) or Vehicular Ah Hoc Network (VANET), the devices are prone to fault, failures and attacks [112,116,117]. Several techniques exist to detect deceptive nodes, but it requires some time and often needs centralized monitoring capabilities. To our knowledge, no work assumed transmission of corrupted predictions to the aggregator node, nor solution in case of unavailability. It is equivalent to a completely decentralized P2P network where there is no leader and corrupted peers can participate to the classification task.

The other branch of EL is *ensemble clustering* (EClus) [118] that combines clustering algorithms. There is no way to know if a partitioning is *good* because there is no ground truth. Therefore, ECLus algorithms must simultaneously deal with the good and bad clustering without any other help. Therefore, this class of algorithms will be helpful to deal with corrupted predictions. While clustering is analogous to classification as it assigns labels to items, the approaches shared by EClass and EClus are limited to bagging and weighting [106,107], where weights are assigned at *inference time* allowing to handle corrupted output

more carefully.

Evidence Accumulation Clustering (EAC) [106,107,119–121] is one of the main EClus methods, which has the advantage of not requiring to match labels between the different clustering. The approach starts by gathering the co-clustering frequencies of all possible pairs of items into a co-association (CA) matrix of size $n \times n$ for n items. The similarity matrix obtained is then re-clustered to obtain the final partitioning. A hierarchical algorithm is often used [119,120] as it allows the final user to decide on the clustering granularity. The major drawback of EAC is the complexity and the memory footprint because of the $n \times n$ matrix. Using *Single-Linkage* (SL) hierarchical clustering [120], the authors proposed to limit the CA matrix to the $k = 20$ nearest neighbors, as SL does not consider distant neighbors. Depending on the dataset, the approach led to better results than other clustering methods exploiting the full matrix while reducing the overall computational complexity.

We inspired ourselves from the EAC methods and the nearest-neighbors trick to deal with noise and improve accuracy. Nonetheless, our approach differs significantly from it as the CA weights are exploited to refine labels rather than clustered to obtain the final classification. The approach will be detailed in the following section.

6.3 Label Refinement with Implicit Boundary Learning

In this section, we will detail our proposed EClass method exploiting the EAC method. The task is to classify an unlabeled dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ with $n = |X|$ elements. The ensemble is a group $\mathcal{A} = \left\{A^{(1)}, A^{(2)}, ..., A^{(k_p)}\right\}$ of k_p classifiers, called sometimes *peers*. The way they are obtained impacts the ensemble accuracy – as in any other EClass methods – but does not impact the overall process. The prediction made by the peer p is denoted $\hat{Y}^{(p)} = A^{(p)}(X) \in \left(\mathcal{L}^{(p)}\right)^n$, where $\mathcal{L}^{(p)}$ is the set of classes that p can distinguish, possibly different from the other classifiers. We will discuss later about this particularity.

The goal of the proposed approach is to rectify the label of an item based on its neighborhood. In general, a classifier makes errors on items close to a class boundary. A peer will collect the peers' opinion to know if two items are on the same side of the class boundary. These weights will be used to rectify uncorrect labels using a weighted voter model. The following paragraphs describe the process and explain the motivation of the different choices.

6.3.1 Gathering Co-Association Matrices

The initial prediction $\hat{Y}^{(p)}$ of the peer p is transformed into the local CA matrix $M^{(p)}$, where $M^{(p)}(\mathbf{x}, \mathbf{x}') = 1$ if $A^{(p)}(\mathbf{x}) = A^{(p)}(\mathbf{x}')$ else 0. Only the pairs $M(\mathbf{x}, \mathbf{x}')$ concerning the k-nearest neighbors (k-NN) of $\mathbf{x} \in X$ (denoted $\mathcal{N}_k(\mathbf{x})$, with $\mathbf{x} \notin \mathcal{N}_k(\mathbf{x})$) are computed. As mentioned in [120], this trick reduces the memory footprint and computational cost from $\mathcal{O}(n^2)$ to $\mathcal{O}(kn)$.

After prediction, peers exchange their results, allowing them to compute locally the average

CA matrix $\mathcal{M}$:

$$\mathcal{M}(\mathbf{x}, \mathbf{x}') = \frac{1}{k_p} \sum_{p=1}^{k_p} M^{(p)}(\mathbf{x}, \mathbf{x}') \tag{6.1}$$

Peers may get different results of $\mathcal{M}$ as a perfect communication model is not assumed here. Some peers may not receive the results from all of their peers, or may get truncated messages depending on the network stability. They replace in that case $\frac{1}{k_p}$ by the number of message received for a particular pair $(\mathbf{x}, \mathbf{x}')$. Unless very few messages are received by a peer, this would not impact the outcome of the process.

6.3.1.0.1 Nearest Pairs: The computation of $\mathcal{M}$ for the closest pairs only is motivated by the idea that NN are likely to belong to the same class unless near a class boundary. In a multi-class classification problem, negative information – indicating that two items belong to different classes – is less informative than positive information. For example, for d classes and two items, there are $d \times (d-1)$ assignment possibilities for negative evidence while only d for positive ones. With that many possibilities, it is unlikely to find the correct classes by chance using negative information. Additionally, for an item in a class representing $100 \times \alpha\%$ of the dataset, $100 \times (1-\alpha)\%$ of the pairs would be 0, leading to many unnecessary data as α shrinks with the number of classes and the class imbalance.

6.3.1.0.2 Class Matches: The use of pairs of co-association makes it possible to dispense with the definition of peer classes $\mathcal{L}^{(p)}$. $\mathcal{M}$ gather the number of times two items are classified together, regardless of the possible labeling errors or granularity level. The EAC approach allows combining classification to clustering algorithms as they both produce labeling, one following a ground-truth definition, the other defined from scratch. Nevertheless, the combination of algorithms with different labels definitions assumes that most of the boundaries are common to the different classes/clusters

As an example, suppose we have three classifiers with different goals, one classifier focusing on *animals* [dog, wolf, cat, lion, frog], another on *sizes* [small, medium, large], and the last on *colors* [white, gray, black, green]. The *animal* and *size* classifiers are compatible as one size corresponds to a unique set of animals (small = {frog}, medium = {dog, cat}, large = {wolf, lion}). Therefore, all the *size* boundaries are preserved by the *animal* classifiers, with some additional. However, the *animal* and *color* classifiers share only the boundary between frog and the other animals which corresponds to the boundary between green and the other colors. Therefore, these two classifiers are weakly compatible.

To obtain compatible classifiers, classes must be derived from a primary set of classes, as in *Error Correcting Output Code* (ECOC) [122] where classes are grouped into two groups to train a binary classifier on it. Another option is available if classes organized into a hierarchical ontology. In this case, high and low-level classes could be combined as several low-level classes derived from a single high-level class. Therefore, the high-level class boundary is preserved within the low-level classes. However, the combination of algorithms with different

class definitions impacts the strength of $\mathcal{M}$ as the ensemble boundaries are less pronounced. This would have almost no impact on the high-level classifiers as low-level ones share their boundaries. Still, low-level classifiers would be as not all their boundaries are not matched by higher-level classifiers.

6.3.1.0.3 Exchanging Pairs: Depending on the network quality and the data privacy preferences, a peer may prefer to exchange its CA matrix or its raw predictions.

The raw predicitions require $8 \times n$ bits per message if a label is encoded over an octet in terms of *bandwidth requirements*. When sending the binary CA matrix, it requires $k \times n$ bits per message. In both cases, we assume that all peers have the same set of items with the same indexing; therefore the features and index do not need to be exchanged. The value of k in our setups is relatively small (≈ 10); therefore one or the other option is equivalent in size.

In an *insecure environment*, peers may prefer not to exchange their raw prediction as an eavesdropper would get roughly labeled data for free. With a CA matrix, the eavesdropper can only recover a partial clustering, limiting the amount of information leakage. In addition, the encryption scheme [123] can be used in binary form, allowing everyone to get the total number of votes for each item without knowing what peers are voting for.

6.3.2 Label Refinement Phase

The refinement phase exploits $\mathcal{M}$ to adjust the initial predictions $\{\hat{Y}^{(p)}\}_{p=1:k_p}$. This step is done locally where each peer will refine its own prediction $\hat{Y}^{(p)}$, without any communication. The initial classification $y_0^{(p)}(\mathbf{x}) = A^{(p)}(\mathbf{x})$ is updated using the following equation:

$$y_{t+1}^{(p)}(\mathbf{x}) = \arg\max_{\ell \in \mathcal{L}^{(p)}} \sum_{\mathbf{x}' \in \mathcal{N}_k(\mathbf{x})} \mathcal{M}(\mathbf{x}, \mathbf{x}') \delta \left(y_t^{(p)}(\mathbf{x}'), \ell \right) \tag{6.2}$$

In other words, $y^{(p)}(\mathbf{x})$ is updated with a weighted voter model, using $\mathcal{M}$ to adjust the neighbors' label contribution. The update process is repeated several times until labels do not change significantly from their previous estimation.

The process can be seen as a horizontal voter model, where instead of looking at all superposed predictions $\{A^{(p)}(\mathbf{x})\}_{p=1:k_p}$, item's nearest neighbors' label $\{A^{(p)}(\mathbf{x}')\}_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})}$ are taken into account. By re-using its prediction, the peer prevents itself from label contamination from noisy or malicious peers as only the weights can be altered but not the labels.

The weights in $\mathcal{M}$ reflect the probability of two items of being in the same class, which indirectly encodes the ensemble's boundary location as it concerns the nearest neighbors. Near a boundary, items on the same side have high weights, whereas items from the other side have low weights. Therefore, only the most relevant items on the same boundary side would contribute positively.

If a boundary in $\mathcal{M}$ does not pre-exist in $M^{(p)}$, nothing would change p's prediction near the

boundary as an item is surrounded by items with the same label. This case occurs when a classifier has a label definition different from the group, which explains why we can combine classifiers with different objectives without issue.

The results of this weighting scheme differ from the k-NN outcome, where all items contribute equally or differently as in the wk-NN version. When density is not homogeneous, one class may contribute more than another because items are easier to find in the neighborhood. Therefore, items would be misclassified, whereas in our approach, relevant items are identified, allowing to rectify labels independently of the density.

6.3.3 Algorithm Complexity

The complexity of the proposed approach is decomposed as follow. The prediction step is at least in $\mathcal{O}(n)$ per peer. Next, the matrices $\{M^{(p)}\}_{p=1:k_p}$ are each obtained in $\mathcal{O}(kn \log n)$ to search for the k-NN and issue the $n \times k$ matrix. The averaging of the matrices $\{M^{(p)}\}_{p=1:k_p}$ into $\mathcal{M}$ is made in $\mathcal{O}(knk_p)$. Last, the refinement step costs $\mathcal{O}(knT)$ for T iterations. In total, each peer has a complexity of $\mathcal{O}((\log n + k_p + T)kn)$, which is larger than other ensemble methods (like boosting with $\mathcal{O}(k_p n)$), but the cost is close to be linear in n. The total cost is quadratic with the number of peers as they each need to average k_p matrices into $\mathcal{M}$. This cost can be significantly reduced using *Gossip* approaches by performing local averaging.

6.4 Experimental Setup

6.4.1 Datasets

We selected four multi-class datasets from the UCI machine learning repository [124] with a sufficient number of instances ($\sim$ 1.000 per class).

A large number of instances allows us to train many classifiers on disjoint item sets while preserving a large part for testing. Some of these datasets were already split into the *training* and *testing* part. We did not keep this split and merged the two sets to obtain a larger experiment testing set.

We tested our approach over multi-class datasets because binary problems are easier to solve than multi-class problems. Multi-class problems have a lower accuracy baseline, therefore a larger margin for improvement. In our proposed approach, we used only positive evidences. In a binary problem, negative evidence is equivalently informative as there are only two class possibilities. For this particular case, $\mathcal{M}$ could be used differently to improve accuracy.

Table 6.1 summarizes the characteristics of the different datasets. The table gathers the number of samples n, the number of classes $|\mathcal{L}|$, and the proportion of the less and most represented classes MinClass and MaxClass respectively. The datasets are correctly balanced, with enough samples in the lowest class, so we do not need to care about class unbalance. We pre-processed all datasets the same way, using z-normalization to give an equal contribution

Table 6.1: Datasets description.

Dataset	n	$\lvert\mathcal{L}\rvert$	MinClass	MaxClass
DryBean	13.611	7	3.83%	26.05%
PenDigit	10.992	10	9.59%	10.40%
Statlog	6.435	6	9.73%	23.82%
USPS	9.298	10	7.61%	16.70%

to each feature.

6.4.2 Weak Learner Setup

In our experiments, we will test the influence of the number of classification peers in the system. We want to train them on a disjoint subset of data to ensure they all are different. As the datasets are limited in size, we selected the k-NN neighbors classifier for simplicity and the low training data requirements. To avoid class imbalance, each classifier's training was composed of d items from each class. d was set to 3 leading to 30 items per classifiers for a dataset with 10 classes like *USPS* and *PenDigit*.

The *USPS* has the lowest ratio $\frac{n}{|\mathcal{L}|}$. In an ensemble with 20 classifiers, it uses 600 items for training, representing less than 7% of the total dataset size. For the *DryBean* dataset with the largest ratio, 3 % of the dataset would be used in the largest training condition. This places the experiment under weakly supervised learning conditions, letting more space for accuracy improvement.

6.5 Results

In this section, we present the results of the experiments comparing our ensemble method – denoted LR for *Label Refinement* – to the simple voter model – denoted VM – using the same ensemble of classifiers. As the LR method is performed locally, not all the different peers obtain the same results. The different predictions can be centralized and aggregated using a voter model. This third possibility is denoted $LR + VM$ in the experiments.

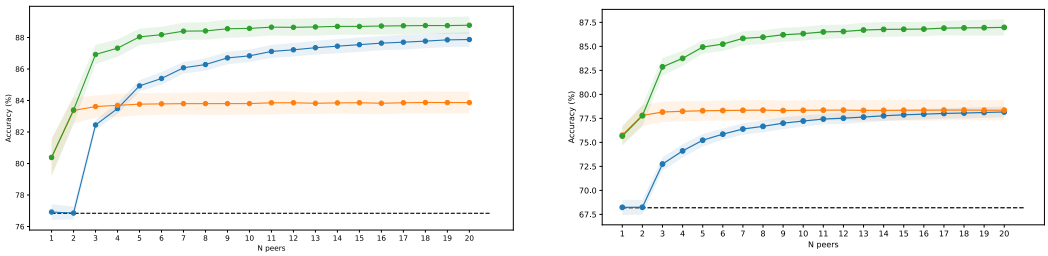
6.5.1 Accuracy Improvement under Stable Conditions

The first experiment explores the impact of the ensemble's size on accuracy. In this setup, the peers are assumed to be trustful and communication perfect, such as the matrices $M^{(p)}$ are exchanged without errors. Fig. 6.1 displays the results of the three different setups and the baseline corresponding to a classifier alone.

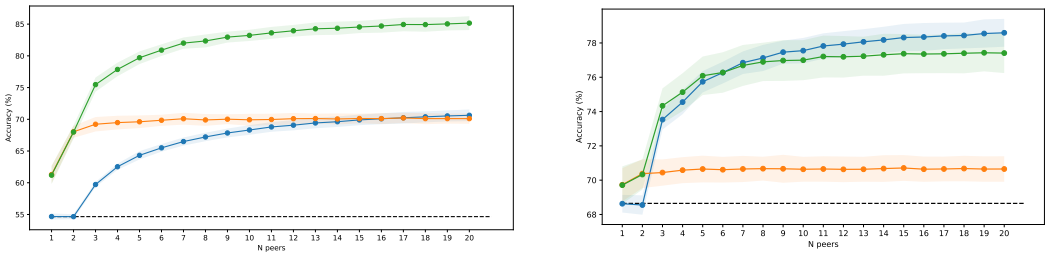
As expected, all ensemble methods are more accurate than the average classifier. The accuracy gain of each method varies from one dataset to another, regardless of the number of classes, number of test samples or the base accuracy.

For all approaches, there is a point for 1 and 2 peers. In the case of the voter model, the results correspond to the average learner accuracy as the scheme cannot help in this configuration. For LR and $LR + VM$, the first point corresponds to the mean accuracy of a learner p refining its prediction with a binary matrix $M^{(p')}$ from another peer p' without combining it with its own. For 2 peers, the internal matrix $M^{(p)}$ is combined with an external one $M^{(p')}$. The LR approach led to accuracy gain for all datasets in this configuration.

Compared to the voter model, the label refinement approach is quickly beatten over the *Drybean* and *StatLog* datasets. However, it is more competitive over the *PenDigit* and *USPS* datasets where the voter model can only reach the same accuracy after gathering 15 learners



(a) DryBean (b) PenDigit



(c) StatLog (d) USPS

Figure 6.1: Accuracy evolution varying with the number of peers. Dashed line: average accuracy of a classifier; orange: LR; green line: $LR + VM$; blue line: VM. Shaded areas correspond to the standard deviation of the mean accuracy over 50 trials.

in the ensemble. The accuracy gain obtained with LR is quite stable after gathering just a few peers. 5 peers are enough to reach the maximal accuracy gain with LR. In comparison, the accuracy of an ensemble using 20 peers can still improve by adding peers under the voter model.

When applying a voter model *after* the label refinement step, an additional gain of accuracy is observed. The gain is non-negligible and allows better accuracy with $LR + VM$ over three of the four datasets with a large margin. While the LR accuracy is stable after gathering 5 peers, the combination of LR to VM is beneficial as $LR + VM$ can still improve accuracy by adding more peers. Nonetheless, it requires another communication phase to collect all the refined predictions.

6.5.2 Resilience to Output Corruption

This second experiment simulates an environment where *output* labels are corrupted at random. The corruption could happen in the peer memory or during transmission. In the real world, it could be materialized by a truncation of the results (one part is unreadable or non-received), or by the addition of noise (some labels are flipped). A noise level α corresponds to the replacement of $\alpha\%$ of the labels by a random label from $\mathcal{L}$. A deletion of $\alpha\%$ of the input could be considered the same way, as missing data are inputted with random labels, to the difference that the position of the incorrect label is known.

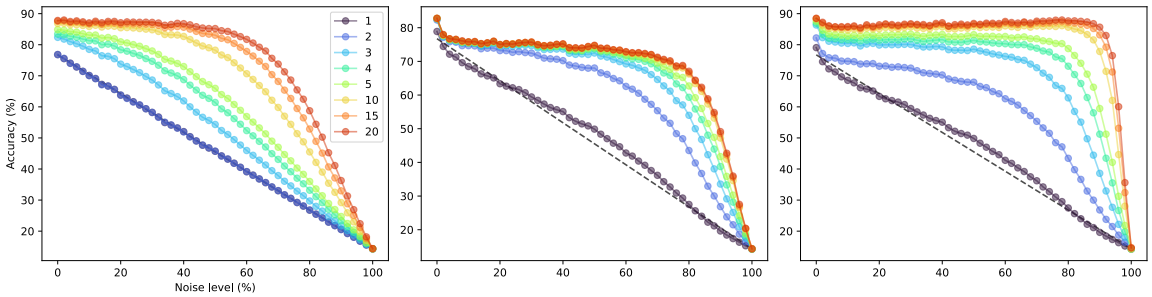
In this configuration, we compare the model's ability to recover the true labels $\mathcal{Y}$ using

only the corrupted labels $\{\hat{Y}_c^{(p)}\}_{p=1:k_p}$. The $\{M_c^{(p)}\}_{p=1:k_p}$ matrices are also derived from the *corrupted* labels and the refinement step starts with the peer's corrupted labels $\hat{Y}_c^{(p)}$. The results of the three different methods over the four datasets are displayed in Fig. 6.2.

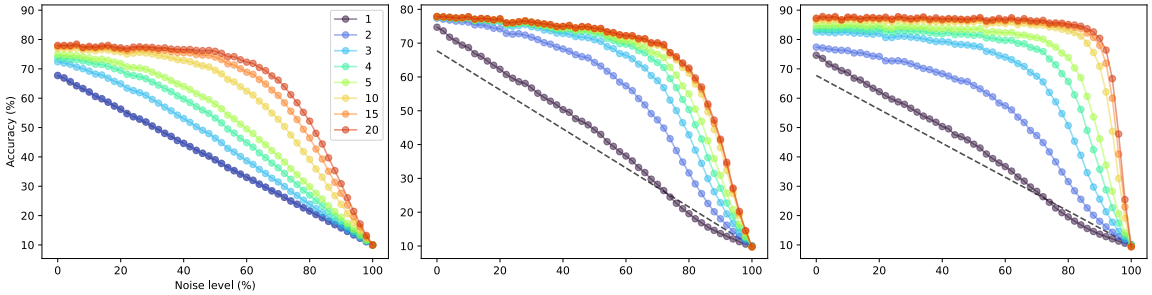
The line corresponding to an ensemble of size 1 corresponds – as in the first experiment – to the average accuracy of corrupted peers in the case of the VM, and the accuracy of a peer refining its corrupted output $\hat{\mathcal{Y}}_c^{(p)}$ with a received binary matrix $M_c^{(p')}$. When increasing the noise level, the VM accuracy decreases linearly. It is almost equal to $\text{acc}(\alpha) = (\text{acc}_0 - |\mathcal{L}|^{-1}).(1-\alpha) + |\mathcal{L}|^{-1}$, where acc_0 is the accuracy without noise, and $|\mathcal{L}|^{-1}$ the probability to select a correct label at random. The dashed line corresponds to this equation to allow an easier comparison of the different methods.

The methods do not have the same response to an increase in noise. VM requires much more peers to obtain a resilience equivalent to LR. A change from 1 to 5 learners offers limited improvement for VM, while in the case of LR, an ensemble with 5 learners is almost as stable as an ensemble of 20 learners. The accuracy of LR using one external matrix M_c for refinement does not offer resilience, as depending on the dataset and the noise level the accuracy is a little bit greater or lower than the baseline accuracy. However, moving to the use of 2 matrices M_c, the resilience of LR surpasses the voter model. $LR + VM$ is even more resilient than the two other methods with very flat horizontal curves

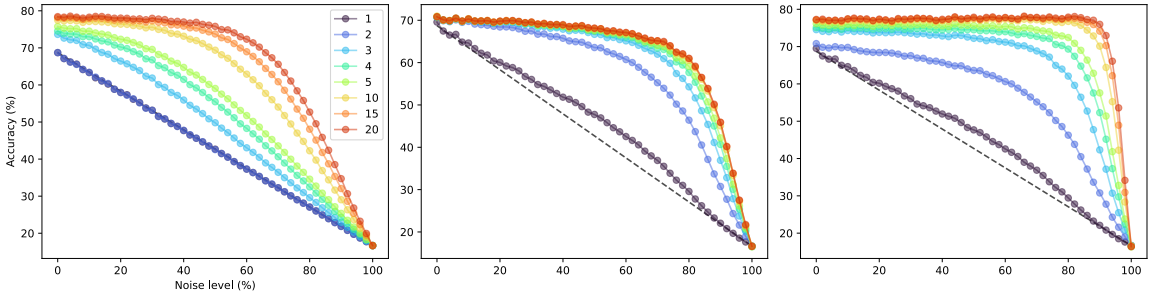
When looking at the red curves corresponding to ensembles of size 20, there are two observable domains: *resilience* and *fragility*. In the *resilience* domain, an increase of noise leads to a small decrease of accuracy, while in the *fragility* domain this increase leads to an important change of accuracy. The transition between the two domains depends on the method used and the ensemble size. The transition occurs around 65% of noise for VM, nearby 80% for LR, and 90% for $LR + VM$. Therefore, the label refinement approach and its variant lead to more resilience than the voter model with the same ensemble.



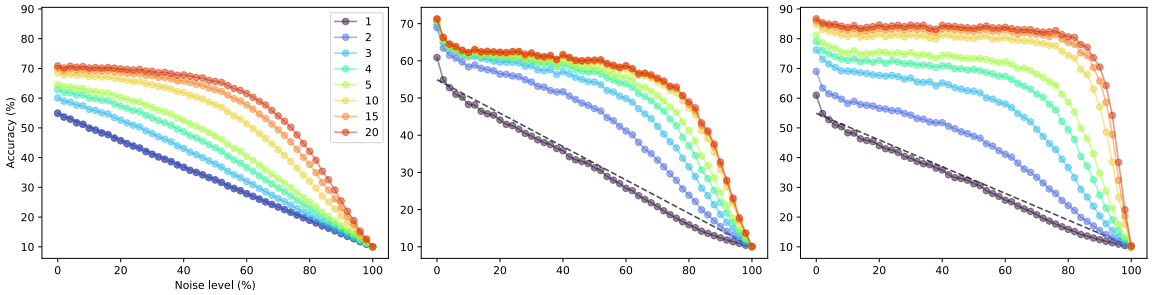
(a) DryBean



(b) PenDigit



(c) StatLog



(d) USPS

Figure 6.2: Resilience to output corruption under increasing level of noise, with boundary parameter $k = 10$. For each dataset, the ensemble models are ordered as follow: left: VM, middle: LR, right: $LR + VM$. Each curve corresponds to a particular ensemble size; the number of peers is indicated in the first figure's legend. The dashed line corresponds to the average corrupted learner accuracy. For VM, the curves corresponding to the ensemble of sizes 1 and 2 overlap with the dashed line. The noise is increased by step of 2%, from 0% to 100%. Each experiment has been repeated 25 times.

6.6 Boundary Size Influence

In the previous experiments, the boundary parameter k controlling the number of nearest neighbors was set to $k = 10$. This parameter is important as it controls the number of items that can induce a label change. Therefore, the third experiment studies the impact of k over the resilience behavior for a fixed number of peers $k_p = 10$. The results are displayed in Fig. 6.3, and similar results are obtained for the other datasets.

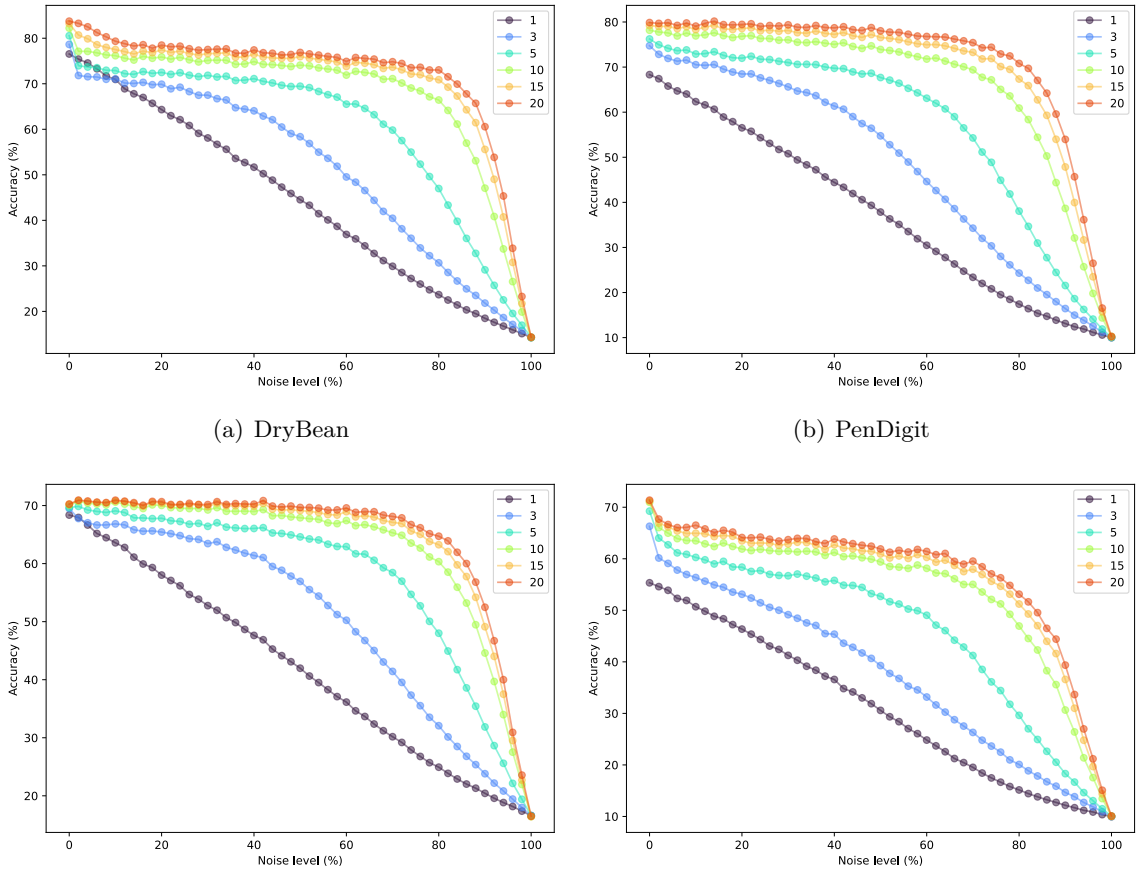
Under the absence of noise, the increase of k has a positive effect on the accuracy of the LR method but is limited in amplitude. In the presence of noise, a larger value of k prevents a premature loss of accuracy.

The configuration with $k = 1$ does not seem to have any impact on the accuracy. In this setup, the only possibility for an item is to inherit from its neighbor's label unless they already have the same label. If $\mathbf{x}$ and $\mathbf{x}'$ are reciprocal neighbors, they will exchange their labels forever. If they are not (i.e., $\mathbf{x} \in \mathcal{N}(\mathbf{x}')$ but $\mathbf{x}' \notin \mathcal{N}(\mathbf{x})$), then $\mathbf{x}$ would take $\mathbf{x}'$'s label. The number of possible changes in this setup is limited, which might explain the small difference from the average classifier accuracy.

The accuracy under noisy condition increase is better recovered with a larger k. The switch from *resilience* to *fragility* also changes from $40 \sim 50\%$ for $k = 3$ to $80 \sim 90$ for $k = 20$. The resilience behavior is easily obtained, with $k = 10$ providing almost the same resilience to noise as $k = 20$. Compared to the curves presented in Fig. 6.2, an increase of the boundary k in LR leads to greater resilience in accuracy than the addition of more peers in the VM. A sufficient k needs to be combined to a sufficient k_p to benefit from the ensemble size and items' neighborhood stability.

The greater resilience to noise with a larger k can be simply explained. In a dense area where all items belong to the same class (before corruption), all nearest items would receive the same weight. Therefore the weighted vote will be equivalent to a normal vote. As k increases, more intact labels would be included in the vote, making the decision more stable. Consequently, items in this area will easily recover their initial labels. The items near the boundary benefit from the same effect because they are surrounded by boundary items that are possibly misclassified. Therefore, k needs to be larger for these area to recover.

When k is small, the density is almost invariant; therefore, all classes might be represented in equal proportion. However, the density can vary from one location to another within the radius when extending the radius. If located in a very dense area, a class will be more numerically represented. The vote in (6.2) is biased by the weights, but also by the number of items that contribute. As an example, if there is one item $\mathbf{x}'$ in the neighborhood of $\mathbf{x}$ belonging to the correct class ℓ with weight $\mathcal{M}(\mathbf{x}, \mathbf{x}') = 1$, but 10 other items of the same incorrect class ℓ_w with weight $\mathcal{M}(\mathbf{x}, \mathbf{x}') = 0.11$, then ℓ_w would win the vote. Therefore, the parameter boundary k must be chosen appropriately to avoid this type of issue. In addition, the process becomes computationally expensive.



(a) DryBean (b) PenDigit

(c) StatLog (d) USPS

Figure 6.3: Resilience to noise for different boundary parameters k, using LR with 10 classifiers in the ensemble.

6.7 Discussion

6.7.1 Instance vs Batch Classification

One major difference between our ensemble method and classical ensemble methods is the input size. Traditionally, the ensemble process one sample at a time, allowing us to predict labels of any sample size without constraint. Our approach is inspired by *Evidence Accumulation Clustering*, where samples are processed by batch. Therefore, it limits the usability of use-cases that do not require real-time answers. Nonetheless, for applications with no real-time constraint, data can be stored into a buffer and classified when the collected data is sufficient.

In the different experiments, we classified all the items left in the test set together. When reducing the test size, the system can become unstable if too few samples are available. In a batch, items can be classified into *core* items that are easily classified by all peers, with most of their nearest neighbors belonging to the same class, and *boundary* items where errors are more frequent. Misclassified core items are easily rectified thanks to their strong neighborhood. However, boundary items rely on core items to get good reference labels,

as other boundary items may not be reliable enough. If the batch size is too small, core areas would be restricted to very few points, and would not be large enough to offer stable references to other items.

We tested over the *DryBean* dataset to reduce the batch size. Before reaching the system instability for $k = 10$ and its 7 classes, the critical size limit was 150 items. Under this size, the mean accuracy decreased while the variance increased.

One way to work on smaller batches is to adapt the boundary parameter k. Decreasing its value may help to keep core areas stable over small batch sizes, while a too large k might destroy core areas due to misclassified boundary items. Another strategy to work on small batches is to classify the batch X together with X' obtain from a database.

The full process is performed on $X_0 \cup X_1$, which would not suffer from possible instabilities.

6.7.2 Combining Clustering with Classification

The use of co-association matrices $\{M^{(p)}\}_{p=1:k_p}$ rather than raw predictions $\{\hat{Y}^{(p)}\}_{p=1:k_p}$ gives the possibility to adapt our approach to clustering or mix clustering with classification results. The combination of the two types of algorithms may help to improve classification accuracy. However, the clustering's benefits might be more limited.

One of the main clustering problems is the selection of the number of clusters. Unfortunately, EClus algorithms often cluster $\mathcal{M}$ using a hierarchical algorithm that circumvents the cluster number problem. The label refinement scheme will help adjust the boundary but will neither help find the best number of clusters.

A direction to explore is how to fuse (or split) clusters using ensemble information. A possibility would be to search for clusters boundaries looking at the local CA matrix $M^{(p)}$ that does not exist in the ensemble matrix $\mathcal{M}$. By detecting weak cluster boundaries, the clusters separated by these boundary could be merged. The same reasoning could be applied for splitting clusters by searching for a strong boundary in $\mathcal{M}$ that does not exist in $M^{(p)}$. This direction can be explored and compared to a more direct clustering of $\mathcal{M}$.

6.7.3 Ensemble of n-ary Classifiers

Mutli-class classifiers can be built by combining binary classifiers, as in the ECOC [122]. Each classifier is trained over 2 classes obtained by splitting the d_0 initial classes at random. This class grouping exploits *class synergies* as two similar classes are easier to distinguish from the other classes when grouped than each alone. The original multi-class would be identified by looking at the ensemble classification binary code obtained.

More generally, d_0 classes can be fused into d_1 classes. However, the transformation of the multi-class prediction to fewer classes can obfuscate the results for privacy reasons. Another motivation is the reduction of the transmitted bit when exchanging the raw predictions.

If $d_0 \gg d_1$, the amount of preserved boundaries is $1 - \frac{1}{d_1}$ (the boundary between two classes merged into the same group is no more identifiable, but still observable for two classes in a distinct group). It is at worse 50% for a binary grouping, impacting $\mathcal{M}$ in these proportion.

Looking at particular class boundaries, a proportion α of the classifiers still see this boundary and would vote something different for each pair of items. On the other hand, a proportion $(1 - \alpha)$ do not see it and would vote 1 for any pairs. Therefore, we could rewrite $\mathcal{M}$ as $\mathcal{M}(\mathbf{x}, \mathbf{x}') = \alpha m(\mathbf{x}, \mathbf{x}') + (1 - \alpha)$ where $m(\mathbf{x}, \mathbf{x}')$ is the average co-association score of the classifiers seeing the boundary. When performing the vote using Eq. (6.2), the part $(1 - \alpha)$ contributes equally for all neighbors leading to a very limited effect on the vote outcome.

6.8 Conclusion

In this article, we proposed a noise-resilient ensemble classification method exploiting the co-classification of nearest items. The system works in two steps: an *exchange phase* where peers communicate their prediction, and a local *label refinement phase* where labels are adjusted using a weighted voter model.

This ensemble approach was tested on four different datasets, where it led to accuracy improvement. Moreover, under the noisy condition, the proposed ensemble method was highly resilient to noise. It allowed it to preserve accuracy with much fewer peers than an ensemble combining its prediction using a voter model. The approach is flexible, as clustering peers can participate in helping classification peers. Additionally, classifiers with different objectives can be combined if some of their class share the same boundaries.

In future works, we will analyze the data factors impacting the accuracy to quantify the possible gain, and investigate how to estimate the boundary parameter k to better adapt to the dataset to classify.

6.9 Contributions

In this thesis, we explored different approaches for finding communities in graphs. We developed several approaches to identify community by paying attention to the two main problems in real-world graphs: sparsity and power-law distribution of nodes' degree. The sparsity problem was addressed by considering the node's neighborhood. The several proposed approaches are presented in the different chapters:

- In chapter 2, sparsity is reduced by exploring indirect neighbors. The node's neighborhood was weighted based on its accessibility from the current node to avoid the problem of very large similarity between two nodes' neighborhood. The similarity between two items was defined as a function of the overlap and the mass distribution over the nodes' neighborhood;
- In chapter 4, the sparsity was reduced by exploiting co-occurrence matrices obtained from both sides of the bipartite graphs, transforming the sparse vector model representation to a dense matrix;
- In chapter 5, nodes were projected on a low dimensional space to circumvent the sparsity problem. In addition, the "distance" between two nodes was measurable on this low dimensional space, permitting to infer relationships' strength between items independently of their initial connectivity.

These methods indirectly transform the initial graph into an equivalent weighted graph with more edges. With a larger number of neighbors whose contribution vary, comparing a node by looking at its neighborhood is more robust due to more accurate information. The second problem concerning power-law distribution was partially addressed, and continues to be a concern in some of the proposed approaches:

- In chapter 2, the directed acyclic graph is asymmetric, where in-degree follows a power-

law while out-degree does not. Therefore, we choose to explore the neighborhood by looking at out-going edges to circumvent the issue;
- In chapter 4, the co-occurrence matrices were normalized to ensure that all nodes exchange the same amount of information with their neighbors independently of their initial predominance. This approach allowed to enrich items' description using their neighborhood without getting overwhelmed by highly connected items;
- In chapter 5, the projection in the low dimensional space limited the number of neighbors and rebalanced neighborhood contribution both for over-frequent and infrequent items mechanically. Nevertheless, too infrequent items tended to create their own cluster. A filtering step prevented their apparition but eliminated a large part of the nodes in some datasets.

By addressing sparsity and power-law issues, each node could be represented with a dense vector allowing comparison with methods designed for traditional tabular data. We explored the usability of these representations by searching natural grouping. Therefore, two co-clustering approaches with different strategies were presented:

- In chapter 4, we proposed a hierarchical co-clustering algorithm, where clusters are forced to grow at the same speed, preventing the apparition of clusters with disproportionate sizes;
- In chapter 5, we proposed a co-embedding process for a bipartite graph, where the representation of one type of node is used to create the embedding for the other nodes. As a result, the items naturally segregate into groups in the low dimensional space that are easily recovered using simple clustering algorithms.

The obtained clustering seemed to group items into coherent thematic clusters, at least for items corresponding to words. Finally, we evaluated the usability of such partitioning for cluster retrieval in chapter 5. The results were conclusive and better than with a clustering obtained using the spectral decomposition method.

Graph visualization is an important topic. Graphs are difficult to project in a 2D space, as configurations with non-crossing edges or keeping edges of equal length rarely exist. A distorted vision of the graph highlighting one of its particularities is often represented instead. We exploited the t-SNE algorithm to represent the graph using different approaches to measure nodes' relatedness:

- In chapter 2, we measured the distance between nodes by looking at their neighborhood's overlap. Compared to the path length which is non-symmetric in a DAG, the proposed distance is symmetric and allows to satisfy better constraints as distances are more nuanced. The embedding let emerged communities of articles referring to the same background knowledge;
- In chapter 3, we developed an extension of the t-SNE algorithm to obtain successive embeddings with clusters sharing the same location. A prior embedding is used to guide items' positioning in a second embedding by applying an additional displacement force.

This update permits to embed a larger graph sliced into several pieces while keeping coherency within the successive frame;

- In chapter 5, the initial objective was to evaluate the usability of a t-SNE embedding to calculate new properties using a low dimensional representation. The final result of the co-embedding process allows visualizing items with similar connectivity, located in the same cluster.

In the last chapter (6), we turned tabular data into a graph by constructing a k-nearest neighbors (k-NN) graph. All classification algorithms in machine learning assume that nearest items belong to the same class unless a boundary separates them. We proposed an ensemble method combining classification results by weighting each edge in the k-NN graph, a weight corresponding to the probability that the two connected items belong to the same class. This method indirectly performed a double voter model and improved accuracy with very few classification results, and tolerated a high level of random noise.

6.9.1 Future Works

In this thesis, we proposed various approaches adapted mainly to directed acyclic graphs and bipartite graphs. There are several topics to explore next.

The proposed algorithms have a complexity often larger than quadratic, restricting their usability to medium-sized graphs. Therefore, the first direction is to study the optimization possibilities. For example, optimization can be done to reduce computations by approximating some values, or using tricks to avoid calculation between unrelated items. Another possibility to reduce the overall computational cost is to work on sampling approaches to obtain a clustering solution over a restricted subset and then assign leftover items to these clusters.

Another orthogonal direction to explore concerns the representation of a DAG. In this particular structure, there is a strong node ordering as there exists no cycle. A node could be represented by a vector based on its reachability from a particular set of nodes. Given the vectors representing the different nodes, clustering algorithms and other algorithms could be applied without considering the local connectivity structure. Nevertheless, using a new representation would require the design of specific algorithms able to measure distance or similarity differently.

The third branch aims to explore the processing of payment graphs. This type of graph combines at the same time a DAG and a 2-mode graph where the monetary flux is exchanged. Each day, there are millions of transactions appended to the existing payment graphs. These large graphs are way too large to be processed with the developed methods, requiring further research on the two first directions. Out of the algorithmic complexity, there is a need to develop appropriate methods to considers this particular structure. The processing and understanding of payment graphs are important because of the rise of dematerialized payment using z credit card, bank transfer, or blockchain. However, malicious activities can happen in all of them; therefore detecting fraud schemes and other activities would be beneficial to

prevent unnecessary harm to many people. Furthermore, out of detecting misbehaving nodes, the high-level understanding of these graphs could help study economic transfers and system redistribution.

Last is to study research fronts for scientific and technical watch. This would be an extension of the 1st and 2nd chapters, where the communities' evolution would be tracked over time to identify trendy topics. Another possibility is to design new clustering algorithms that do not need to project the graph to scale to large graphs. The investment in this direction would help save time on the other research topics by fastening the understanding of the ecosystem and the identification of the most relevant papers.

References

[1] Garfield E 2004 Historiographic mapping of knowledge domains literature *Journal of Information Science* **30** 119–45

[2] Kessler M M 1963 Bibliographic coupling between scientific papers *American Documentation* **14** 10–25

[3] Small H 1973 Co-citation in the scientific literature: A new measure of the relationship between two documents *J. Am. Soc. Inf. Sci.* **24** 265–9

[4] Grauwin S and Jensen P 2011 Mapping scientific institutions *Scientometrics* **89** 943–54

[5] Ding Y 2011 Scientific collaboration and endorsement: Network analysis of coauthorship and citation networks *Journal of informetrics* **5 1** 187–203

[6] Boyack K W, Klavans R and Börner K 2005 Mapping the backbone of science *Scientometrics* **64** 351–74

[7] Boyack K and Klavans R 2010 Co-citation analysis, bibliographic coupling, and direct citation: Which citation approach represents the research front most accurately? *J. Assoc. Inf. Sci. Technol.* **61** 2389–404

[8] Klavans R and Boyack K 2017 Which type of citation analysis generates the most accurate taxonomy of scientific and technical knowledge? *Journal of the Association for Information Science and Technology* **68**

[9] Redner S 1998 How popular is your paper? An empirical study of the citation distribution *The European Physical Journal B - Condensed Matter and Complex Systems* **4** 131–4

[10] Price D 1976 A general theory of bibliometric and other cumulative advantage processes *J. Am. Soc. Inf. Sci.* **27** 292–306

[11] Merton R 1968 The matthew effect in science *Science* **159** 56–63

[12] Allen H, Stanton T, Pietro F D and Moseley G 2013 Social media release increases dissemination of original articles in the clinical pain sciences *PLoS ONE* **8**

[13] Gargouri Y, Hajjem C, Larivière V, Gingras Y, Carr L, Brody T and Harnad S 2010 Self-selected or mandated, open access increases citation impact for higher quality research *PLoS ONE* **5**

[14] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M and others 2016 TensorFlow: A system for large-scale machine learning. *OSDI* vol 16 pp 265–83

[15] Tahamtan I and Bornmann L 2019 What do citation counts measure? An updated review of studies on citations in scientific documents published between 2006 and 2018 *Scientometrics* **121** 1635–84

[16] Kerckhoffs A 1883 La cryptographie militaire *Journal des sciences militaires* **IX** 161–91

[17] Jolliffe I 2005 *Principal component analysis* (John Wiley & Sons, Ltd)

[18] Garfield E and Zuckerman H 1980 Multiple independent discovery and creativity in science *Essays of an Information Scientist* **4** 1979–80

[19] Jarneving B 2007 Bibliographic coupling and its application to research-front and other core documents *J. Informetrics* **1** 287–307

[20] Leydesdorff L and Ràfols I 2009 A global map of science based on the ISI subject categories *J. Assoc. Inf. Sci. Technol.* **60** 348–62

[21] Huang M-H and Chang C-P 2013 Detecting research fronts in OLED field using bibliographic coupling with sliding window *Scientometrics* **98** 1721–44

[22] Fujita K, Kajikawa Y, Mori J and Sakata I 2012 Detecting research fronts using different types of weighted citation networks *2012 Proceedings of PICMET '12: Technology Management for Emerging Technologies* 267–75

[23] Colavizza G, Boyack K, Eck N J van and Waltman L 2018 The closer the better: Similarity of publication pairs at different cocitation levels *Journal of the Association for Information Science and Technology* **69**

[24] Huang M-H, Chiang L and Chen D-Z 2004 Constructing a patent citation map using bibliographic coupling: A study of taiwan's high-tech companies *Scientometrics* **58** 489–506

[25] Yan E and Ding Y 2012 Scholarly network similarities: How bibliographic coupling networks, citation networks, cocitation networks, topical networks, coauthorship networks, and coword networks relate to each other *J. Assoc. Inf. Sci. Technol.* **63** 1313–26

[26] Page L, Brin S, Motwani R and Winograd T 1999 The PageRank citation ranking : Bringing order to the web *WWW 1999*

[27] Tang J, Zhang J, Yao L, Li J, Zhang L and Su Z 2008 ArnetMiner: Extraction and mining of academic social networks *KDD'08* pp 990–8

[28] Maaten L van der and Hinton G 2008 Visualizing data using t-SNE *Journal of Machine Learning Research* **9** 2579–605

[29] Cheng Y 1995 Mean shift, mode seeking, and clustering *IEEE Trans. Pattern Anal. Mach. Intell.* **17** 790–9

[30] Cohen S, Ruppin E and Dror G 2005 Feature selection based on the shapley value. pp 665–70

[31] Hall M A 1999 *Correlation-based feature selection for machine learning* PhD thesis

[32] Masci J, Meier U, Ciresan D and Schmidhuber J 2011 Stacked convolutional auto-encoders for hierarchical feature extraction pp 52–9

[33] Maggipinto M, Masiero C, Beghi A and Susto G A 2018 A convolutional autoencoder approach for feature extraction in virtual metrology *Procedia Manufacturing* **17** 126–33

[34] Kohonen T 2001 *Self-organizing maps* (Berlin: Paperback; Springer)

[35] Tenenbaum J B, Silva V de and Langford J C 2000 A global geometric framework for nonlinear dimensionality reduction *Science* **290** 2319

[36] McInnes L, Healy J and Melville J 2018 UMAP: Uniform manifold approximation and projection for dimension reduction

[37] Kobak D and Berens P 2019 The art of using t-SNE for single-cell transcriptomics *bioRxiv*

[38] Pezzotti N, Mordvintsev A, Höllt T, Lelieveldt B P F, Eisemann E and Vilanova A 2018 Linear tSNE optimization for the web *CoRR* **abs/1805.10817**

[39] Maaten L van der 2009 Learning a parametric embedding by preserving local structure *Proceedings of the twelth international conference on artificial intelligence and statistics* Proceedings of machine learning research vol 5, ed D van Dyk and M Welling (Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR) pp 384–91

[40] Min M R, Guo H and Shen D 2017 Parametric t-distributed stochastic exemplar-centered embedding *CoRR* **abs/1710.05128**

[41] Boytsov A, Fouquet F, Hartmann T and Traon Y L 2017 Visualizing and exploring dynamic high-dimensional datasets with LION-tSNE *CoRR* **abs/1708.04983**

[42] Rauber P E, Falcão A X and Telea A C 2016 Visualizing time-dependent data using dynamic t-SNE *Proceedings of the eurographics / IEEE VGTC conference on visualization: Short papers* EuroVis '16 (Goslar, DEU: Eurographics Association) pp 73–7

[43] Harris C R, Millman K J, Walt S J van der, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith N J, Kern R, Picus M, Hoyer S, Kerkwijk M H van, Brett M, Haldane A, R'ıo J F del, Wiebe M, Peterson P, G'erard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C and Oliphant T E 2020 Array programming with NumPy *Nature* **585** 357–62

[44] Wang K, Shen Z, Huang C, Wu C-H, Eide D, Dong Y, Qian J, Kanakia A, Chen A and Rogahn R 2019 A review of microsoft academic services for science of science studies *Frontiers in Big Data* **2** 45

[45] Weinberg B H 1974 Bibliographic coupling: A review *Information Storage and Retrieval* **10** 189–96

[46] Chavalarias D and Cointet J-P 2009 The reconstruction of science phylogeny

[47] Maaten L van der 2013 Barnes-hut-SNE *ICLR* ed Y Bengio and Y LeCun

[48] Gupta M, Li R, Yin Z and Han J 2010 Survey on social tagging techniques *SIGKDD Explorations* **12** 58–72

[49] Bertin-Mahieux T, Ellis D P W, Whitman B and Lamere P 2011 The million song dataset. *ISMIR* ed A Klapuri and C Leider (University of Miami) pp 591–6

[50] Peters I and Stock W 2010 "Power tags " in information retrieval *Library Hi Tech* **28** 81–93

[51] Banerjee S, Ramanathan K and Gupta A 2007 Clustering short texts using wikipedia pp 787–8

[52] Fellbaum C 1998 *WordNet: An electronic lexical database* (Bradford Books)

[53] Wikipedia 2021 Wikipedia, the free encyclopedia

[54] Patil A 2015 Clustering on uncertain data using kullback leibler divergence measurement based on probability distribution

[55] Hassan-Montero Y and Herrero-Solana V 2006 Improving tag-clouds as visual information retrieval interfaces *InScit2006: International conference on multidisciplinary information sciences and technologies*

[56] Hearst M A and Rosner D K 2008 Tag clouds: Data analysis tool or social signaller? *HICSS* (IEEE Computer Society) p 160

[57] Begelman G, Keller P and Smadja F 2006 Automated tag clustering: Improving search and exploration in the tag space *Proceedings of the collaborative web tagging workshop at the WWW 2006* (Edinburgh, Scotland)

[58] Knautz K, Soubusta S and Stock W 2010 Tag clusters as information retrieval interfaces *Proceedings of the Annual Hawaii International Conference on System Sciences* pp 1–0

[59] Wartena C and Brussee R 2008 Topic detection by clustering keywords pp 54–8

[60] Zhao Q, Rezaei M and Chen H 2012 Keyword clustering for automatic categorization *Proceedings - International Conference on Pattern Recognition* pp 2845–8

[61] Carrasco J J, Fain D, Lang K and Zhukov L 2003 Clustering of bipartite advertiser-keyword graph

[62] Koh S and Chia L-T 2006 Web image clustering with reduced keywords and weighted bipartite spectral graph partitioning vol 4261 pp 880–9

[63] Dhillon I S 2001 Co-clustering documents and words using bipartite spectral graph partitioning pp 269–74

[64] Dhillon I S, Mallela S and Modha D S 2003 Information-theoretic co-clustering *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (New York, NY, USA: ACM) pp 89–98

[65] Knight P A 2008 The sinkhorn-knopp algorithm: Convergence and applications *SIAM J. Matrix Anal. Appl.* **30** 261–75

[66] Saad F, Mohamed O and Al-Qutaish R 2013 Comparison of hierarchical agglomerative algorithms for clustering medical documents *International Journal of Software Engineering and Applications* **3** 1–5

[67] El-Hamdouchi A and Willett P 1989 Comparison of Hierarchic Agglomerative Clustering Methods for Document Retrieval *The Computer Journal* **32** 220–7

[68] Li F, Yin Y, Shi J, Mao X and Shi R 2019 Method of feature reduction in short text classification based on feature clustering *Applied Sciences* **9**

[69] Pelleg D and Moore A 2000 X-means: Extending k-means with efficient estimation of the number of clusters *In proceedings of the 17th international conf. On machine learning* (Morgan Kaufmann) pp 727–34

[70] Sarwar B, Karypis G, Konstan J and Riedl J 2002 Recommender systems for large-scale e-commerce : Scalable neighborhood formation using clustering

[71] Conner J H 1999 Clustering items for collaborative filtering

[72] George T and Merugu S 2005 A scalable collaborative filtering framework based on co-clustering *ICDM'05* 4 pp

[73] Vdorhees E M 1985 The cluster hypothesis revisited *SIGIR Forum* **51** 35–43

[74] Chen Y, Wang J Z and Krovetz R 2005 CLUE: Cluster-based retrieval of images by unsupervised learning *IEEE Transactions on Image Processing* **14** 1187–201

[75] Liu X and Croft W 2004 Cluster-based retrieval using language models *SIGIR '04*

[76] Salton G, Wong A and Yang C 1975 A vector space model for automatic indexing *Commun. ACM* **18** 613–20

[77] Ravindran M and Thanamani A 2015 K-means document clustering using vector space model *Bonfring International Journal of Data Mining* **5** 10–4

[78] Lei L, Qi J and Zheng K 2019 Patent analytics based on feature vector space model: A case of IoT *IEEE Access* **7** 45705–15

[79] Sarıyüce A E and Pinar A 2018 Peeling bipartite networks for dense subgraph discovery *Proceedings of the eleventh ACM international conference on web search and data mining* WSDM '18 (New York, NY, USA: Association for Computing Machinery) pp 504–12

[80] Latapy M, Magnien C and Vecchio N D 2008 Basic notions for the analysis of large two-mode networks *Soc. Networks* **30** 31–48

[81] Arroyo J, Priebe C and Lyzinski V 2020 Graph matching between bipartite and unipartite networks: To collapse, or not to collapse, that is the question *ArXiv* **abs/2002.01648**

[82] Charrad M and Ben Ahmed M 2011 Simultaneous clustering: A survey *Pattern recognition and machine intelligence* ed S O Kuznetsov, D P Mandal, M K Kundu and S K Pal (Berlin, Heidelberg: Springer Berlin Heidelberg) pp 370–5

[83] Kaiser S 2011 Biclustering: Methods, software and application

[84] Zha H, He X, Ding C, Gu M and Simon H 2001 Bipartite graph partitioning and data clustering *CIKM '01*

[85] Nie F, Wang X, Deng C and Huang H 2017 Learning a structured optimal bipartite graph for co-clustering *Advances in neural information processing systems* vol 30, ed I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan and R Garnett (Curran Associates, Inc.)

[86] Chen Y, Wang L and Dong M 2010 Non-negative matrix factorization for semisupervised heterogeneous data coclustering *IEEE Transactions on Knowledge and Data Engineering* **22** 1459–74

[87] Du R, Kuang D, Drake B L and Park H 2017 DC-NMF: Nonnegative matrix factorization based on divide-and-conquer for fast clustering and topic modeling *Journal of Global Optimization* **68** 777–98

[88] Selosse M, Jacques J and Biernacki C 2020 Model-based co-clustering for mixed type data *Computational Statistics & Data Analysis* **144** 106866

[89] Gabrilovich E and Markovitch S 2007 Computing semantic relatedness using wikipedia-based explicit semantic analysis *IJCAI*

[90] Tsatsaronis G and Panagiotopoulou V 2009 A generalized vector space model for text retrieval based on semantic relatedness *EACL*

[91] Abbasi R and Staab S 2009 RichVSM: enRiched vector space models for folksonomies *HT '09*

[92] Sidorov G, Gelbukh A, Gómez-Adorno H and Pinto D 2014 Soft similarity and soft cosine measure: Similarity of features in vector space model *Computación y Sistemas* **18**

[93] Benz D, Hotho A, Jäschke R, Krause B, Mitzlaff F, Schmitz C and Stumme G 2010 The social bookmark and publication management system BibSonomy *The VLDB Journal* **19** 849–75

[94] Duygulu P, Barnard K, Freitas J F G de and Forsyth D A 2002 Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary *Computer vision, ECCV 2002* LNCS vol 2353 pp 97–112

[95] Read J, Pfahringer B, Holmes G and Frank E 2011 Classifier chains for multi-label classification *Machine Learning* **85** 333–59

[96] Harper F M and Konstan J A 2015 The MovieLens datasets: History and context *ACM Trans. Interact. Intell. Syst.* **5**

[98] Tsoumakas G, Katakis I and Vlahavas I 2008 Effective and efficient multilabel classification in domains with large number of labels *Proc. ECML/PKDD workshop on mining multidimensional data, antwerp, belgium, MMD08* pp 30–44

[98] Tsoumakas G, Katakis I and Vlahavas I 2008 Effective and efficient multilabel classification in domains with large number of labels *Proc. ECML/PKDD workshop on mining multidimensional data, antwerp, belgium, MMD08* pp 30–44

[99] Lang K 1995 Newsweeder: Learning to filter netnews *Proc. 12th international conference on machine learning* pp 331–9

[100] Joachims T 1998 Text categorization with support vector machines: Learning with many relevant features *ECML*

[101] Huiskes M J and Lew M S 2008 The MIR flickr retrieval evaluation *MIR '08: Proceedings of the 2008 ACM international conference on multimedia information retrieval* (New York, NY, USA: ACM)

[102] Zhou Z 2012 Ensemble methods: Foundations and algorithms

[103] Breiman L 2004 Bagging predictors *Machine Learning* **24** 123–40

[104] Khan Z, Gul A, Perperoglou A, Miftahuddin M, Mahmoud O, Adler W and Lausen B 2020 Ensemble of optimal trees, random forest and random projection ensemble classification *Advances in Data Analysis and Classification* **14** 97–116

[105] Schapire R and Freund Y 2012 Boosting: Foundations and algorithms

[106] Li T and Ding C 2008 Weighted consensus clustering *SDM*

[107] Ren Y-Z, Domeniconi C, Zhang G and Yu G-X 2013 Weighted-object ensemble clustering *IEEE 13th International Conference on Data Mining* 627–36

[108] Margineantu D and Dietterich T G 1997 Pruning adaptive boosting *ICML*

[109] Fan W, Stolfo S and Zhang J 1999 The application of AdaBoost for distributed, scalable and on-line learning *KDD '99*

[110] Ormándi R, Hegedüs I and Jelasity M 2013 Gossip learning with linear models on fully distributed data *Concurrency and Computation: Practice and Experience* **25**

[111] Abualkibash M, ElSayed A and Mahmood A 2013 Highly scalable, parallel and distributed AdaBoost algorithm using light weight threads and web services on a network of multi-core machines *ArXiv* **abs/1306.1467**

[112] Ratasich D, Khalid F, Geissler F, Grosu R, Shafique M and Bartocci E 2019 A roadmap toward the resilient internet of things for cyber-physical systems *IEEE Access* **7** 13260–83

[113] Probst P and Boulesteix A 2017 To tune or not to tune the number of trees in random forest? *J. Mach. Learn. Res.* **18** 181:1–8

[114] Pes B 2019 Ensemble feature selection for high-dimensional data: A stability analysis across multiple domains *Neural Computing and Applications* **32** 5951–73

[115] Breiman L 2001 Random forests *Machine Learning* **45** 5–32

[116] Sen J 2009 A survey on wireless sensor network security *ArXiv* **abs/1011.1529**

[117] Sheikh M S and Liang J 2019 A comprehensive survey on VANET security services in traffic management system *Wirel. Commun. Mob. Comput.* **2019** 2423915:1–23

[118] Strehl A and Ghosh J 2002 Cluster ensembles — a knowledge reuse framework for combining multiple partitions *J. Mach. Learn. Res.* **3** 583–617

[119] Galdi P, Napolitano F and Tagliaferri R 2014 Consensus clustering in gene expression *CIBB*

[120] Fred A and Jain A K 2005 Combining multiple clusterings using evidence accumulation *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27** 835–50

[121] Monti S, Tamayo P, Mesirov J and Golub T 2004 Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data *Machine Learning* **52** 91–118

[122] Dietterich T G and Bakiri G 1995 Solving multiclass learning problems via error-correcting output codes *ArXiv* **cs.AI/9501101**

[123] Hao F, Ryan P and Zielinski P 2010 Anonymous voting by two-round public discussion *IET Inf. Secur.* **4** 62–7

[124] Dua D and Graff C 2017 UCI machine learning repository

RÉSUMÉ

L'objet de cette thèse est de proposer des approches nouvelles permettant l'utilisation d'algorithmes d'apprentissage automatique travaillant usuellement des données tabulaires aux graphes. Un graphe est une structure de donnée composée de nœuds reliés entre eux par des liens. Cette structure peut être représentée sous la forme d'une matrice, où chaque connexion entre de noeuds est représentée par une valeur non nulle, permettant une manipulation des données plus facile. Néanmoins, par leurs différences structurelles, la transposition d'un algorithme exploitant des données tabulaire aux graphes ne donne pas les résultats escomptés. Deux caractéristiques rendent cette adaptation difficile: la faible connectivité des noeuds ainsi que la distribution en loi de puissance du degré des nœuds. Ces caractéristiques conduisent toutes les deux à des matrices creuses pauvre en information tout en nécessitant beaucoup de mémoire de stockage. Dans ces travaux, nous proposons plusieurs manières de prendre en compte ces différences pour deux types de graphes particuliers. Dans la première partie, nous nous intéressons aux graphes de citations et à leur représentation dans l'optique de la veille technologique, tandis que la seconde partie s'adresse aux graphes bipartites utilisés principalement par les systèmes de recommandation. Ces adaptations permettent la réalisation de taches usuelles en apprentissage automatique, telle que le partitionnement et la visualisation des données. Pour le cas des graphes bipartites, des algorithmes spécifiques de co-partitionnement sont proposés pour la segmentation conjointe des deux parties. La troisième partie prend un revers différent. La méthode développée exploite le graphe des k plus proche voisins construit à partir des données tabulaires afin de corriger des erreurs de classifications. Les différentes méthodes développées utilisent diverses approches pour emmagasiner plus d'information dans un vecteur par rapport à l'encodage binaire habituel, permettant de travailler les graphes avec des algorithmes usuel d'apprentissage automatique.

MOTS CLÉS

Graphes, Réseaux, Citation, Bipartite, Partitionnement, Partitionnement join, Détection de communautés, Projection, Visualisation, Apprentissage automatique

ABSTRACT

This thesis proposes new approaches to process graph using machine learning algorithms designed for tabular data. A graph is a data structure made of nodes linked to each others by edges. This structure can be represented under a matrix form where the connection between two nodes is represented by a non-zero value, simplifying the manipulation of the data. Nonetheless, the transposition of an algorithm adapted to tabular data to graphs would not give the expected results because of the structural differences. Two characteristics make the transposition difficult: the low nodes' connectivity and the power-law distribution of nodes' degree. These two characteristics both lead to sparse matrices with low information content while requiring a large memory. In this work, we propose several methods that considers these two graph's specificities. In the first part, we focus on citation graphs which belong to the directed acyclic graph category and can be exploited for technical watch, while the second part is dedicated to bipartite graphs mainly use by recommender systems. These adaptation permit the achievement of usual machine learning tasks, such as clustering and data visualization. Specific co-clustering algorithms were designed to segment jointly each side of a bipartite graph and identify groups of similar nodes. The third part approaches graphs from a different perspective. The developed approach exploits the k nearest neighbors graph built from the tabular data to help correcting classification errors. These different methods use diverse methods to embed more information in a vector compared to the usual binary encoding, allowing to process graphs with usual machine learning algorithm.

KEYWORDS

Graphs, Networks, Citation, Bipartite, Clustering, Co-clustering, Community detection, Embedding, Visualization, Machine Learning