Exploring Topic Evolution in Large Scientific Archives with Pivot Graphs

Ke Li

SORBONNE UNIVERSITÉ

LABORATOIRE INFORMATIQUE PARIS 6

DOCTORAL THESIS

Discipline: Informatique

Exploring Topic Evolution in Large Scientific Archives with Pivot Graphs

Ke LI

Rapporteurs:	Nicolas TRAVERS	Professeur, HDR, ESILV
	Mirian HALFELD-FERRARI	Professeur des Universités, Université d'Orléans
Examinatrices:	Nathalie AUSSENAC-GILLES	Directrice de Recherche CNRS, Toulouse III
	Clémence MAGNIEN	Directrice de Recherche CNRS, Sorbonne Université
Directeurs:	Bernd AMANN	Professeur des Universités, Sorbonne Université
	Hubert NAACKE	Maître de Conférences, Sorbonne Université

June 22*nd*, 2021

DECLARATION

I hereby declare that the work in this dissertation entitled "**Exploring Topic Evolution in Large Scientific Archives with Pivot Graphs**" is my own original work and has not been submitted for another examination or assignment, either wholly or excerpts thereof. Furthermore, I confirm that I have acknowledged the work of others by providing detailed references of said work.

Ke LI

June 22*nd*, 2021

Acknowledgments

First and foremost, I would like to thank my wife **Meng**, whose unconditional love, company and support has been a bright light that shines on me, even in the hardest hours. Her advices and encouragement have made me a better researcher, a better husband, and a better person. I feel so happy and lucky for so many unforgettable and important moments that we have experienced together. The world is big and we still have a long way to go to accomplish our goals about exploring the world. I will be proud to tell her that no matter what the future life is like, I will give her all the support and let her bravely keep going.

I would like to thank **my parents** who understood me and supported me in the tough but exciting path to pursue a Ph.D. degree in Computer Science. They taught me so many things that I have never learnt from the school. They taught me that I should be enthusiastic, passionate and also patient to what I really want to do, which makes me maintain a good rhythm in my pursuit of this thesis. I am so grateful for their sacrifices and efforts that make me able to find strength and work through problems and obstacles that come my way.

Then I would like to express my most sincere gratitude to my supervisors, **Mr. Bernd AMANN** and **Mr. Hubert NAACKE**. I am so grateful for their guidance of my research over the past few years. They have given me not only expertise in the field, but also instruction on how to become a qualified researcher. And, throughout my hard work, I am very grateful for their inspiration, insights, wisdom and understanding. Their enthusiasm and passion has been a great source of motivation and moved me to accomplish many things I could not have dreamed possible.

I owe a deep sense of gratitude to **Mr. Olivier CURÉ** and **Mr. Maximilien DANISCH** for helpful discussions with me, especially for their great interest and guidance at every important stage of my research. Their timely inspirations and suggestions, enthusiasm and dynamism have enabled me to make my thesis better.

I also have many thanks to my team members of LIP6, **Amine**, **Anne**, **Camélia**, **Fatma**, **Rutian**, and **Stéphane**, with whom I have learnt a lot. I thank for their hospitality and friendliness as well.

Finally, I would like to thank my friend, **Xiangnan** who has given me some important and constructive advices on my research and life.

Abstract

There is an increasing demand for practical tools to explore the evolution of scientific research published in bibliographic archives such as the Web of Science (WoS), arXiv, PubMed or ISTEX. Revealing meaningful evolution patterns from these document archives has many applications and can be extended to synthesize narratives from datasets across multiple domains, including news stories, research papers, legal cases and works of literature. In this thesis, we propose a data model and query language for the visualization and exploration of topic evolution graphs. Our model is independent of a particular topic extraction and alignment method and proposes a set of semantic and structural metrics for characterizing and filtering meaningful topic evolution patterns. These metrics are particularly useful for the visualization and the exploration of large topic evolution graphs. We also present a prototype implementation of our model on top of Apache Spark and experimental results obtained for four real-world document archives.

Résumé

Il existe une demande croissante d'outils pratiques pour explorer l'évolution de la recherche scientifique publiée dans des archives bibliographiques telles que le Web of Science (WoS), arXiv, PubMed ou ISTEX. La révélation de modèles d'évolution significatifs à partir de ces archives documentaires a de nombreuses applications et peut être étendue pour synthétiser des récits à partir d'ensembles de données dans plusieurs domaines, y compris les nouvelles, les articles de recherche, les cas juridiques et les œuvres littéraires. Dans cette thèse, nous proposons un modèle de données et un langage d'interrogation pour la visualisation et l'exploration de graphes d'évolution de sujets. Notre modèle est indépendant d'une méthode particulière d'extraction et d'alignement de sujets et propose un ensemble de métriques sémantiques et structurelles pour caractériser et filtrer des modèles d'évolution de sujets significatifs. Ces métriques sont particulièrement utiles pour la visualisation et l'exploration de grands graphes d'évolution de sujets. Nous présentons également un prototype d'implémentation de notre modèle sur Apache Spark et les résultats expérimentaux obtenus pour quatre archives de documents du monde réel.

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 Modeling Science Evolution

The evolution of science and technology is an important indicator for the industrial and economic progress of our society. The study of science evolution can help (1) scientists who want to position themselves in their field [1], (2) policy makers who want to spot emerging fields, foster innovation and get key indicators to assist them in decision-making processes [2, 3], (3) industrialists who have to find their way through the scientific production and evaluate the potential for innovation and technological transfer [4, 5], (4) librarians who need to propose classifications of documents [6, 7], and (5) philosophers and historians of science to test their theories with data [8, 9]. Within this context, there is an increasing demand from experts for practical tools that assist them to extract information about the scientific progress and technological innovations published in bibliographic archives such as the Web of Science (WoS)[1], arXiv[2], PubMed [10] or ISTEX[3].

The evolution of scientific archives can broadly be studied by adopting a cognitive view or a social view on the evolution dynamics. The *cognitive view* of scientific archive evolution emphasizes the shared knowledge and the change of ideas present in the content of document [11, 12], whereas the *social view* takes account of authorship information and social interactions represented, for example, in co-authorship and citation graphs [13, 14, 15]. There also exist methods which combine both views to study science evolution [16, 17, 18].

In this thesis, we adopt the cognitive view and analyze *science evolution patterns* extracted from the textual document contents (title, abstract and main contents). The choice of the cognitive view reduces the number of analysis features, but it also decreases the "social" bias and makes it easier to detect possible interactions between scientific ideas and contributions, independently of any particular scientific community.

Scientific ideas, concepts and contributions published in textual documents at different periods of time can be represented by *topics* extracted from the document content. By connecting or *aligning topics* from different periods, for example by using their similarities, it is possible to generate a structured representation of the research progress in scientific document archives. Topic alignments define a *topic evolution network* or *phylomemy* [19]. The notion of phylomemy or phylomemetic network is inspired from the notion of *phylogenetic* tree representing the characteristics and evolution of species and derived from the genes of their members. Phylomemetic networks track the evolution of science by identifying and analyzing *science evolution patterns* like the emergence and decay of research topics or the split of one research topic into several subtopics, etc.

Example 1 *Figure 1.1 shows two snippets of a single topic evolution network (phylomemy) extracted from the arXiv*[4] *corpus. The graph covers the period between 2000 and 2006 decomposed into three* 3*-year time periods overlapping by one year. Each topic is represented by a rectangle containing the top-*10 *weighted topic terms obtained by a NLP document pre-processing and topic extraction workflow. The terms in each topic can be classified into four disjoint categories with respect to their temporal evolution. Emerging terms are shown in green, decaying term boxes are colored in red, stable terms*

[1] https://clarivate.com/webofsciencegroup/solutions/web-of-science/
[2] https://arxiv.org/
[3] https://www.istex.fr/
[4] https://arxiv.org/

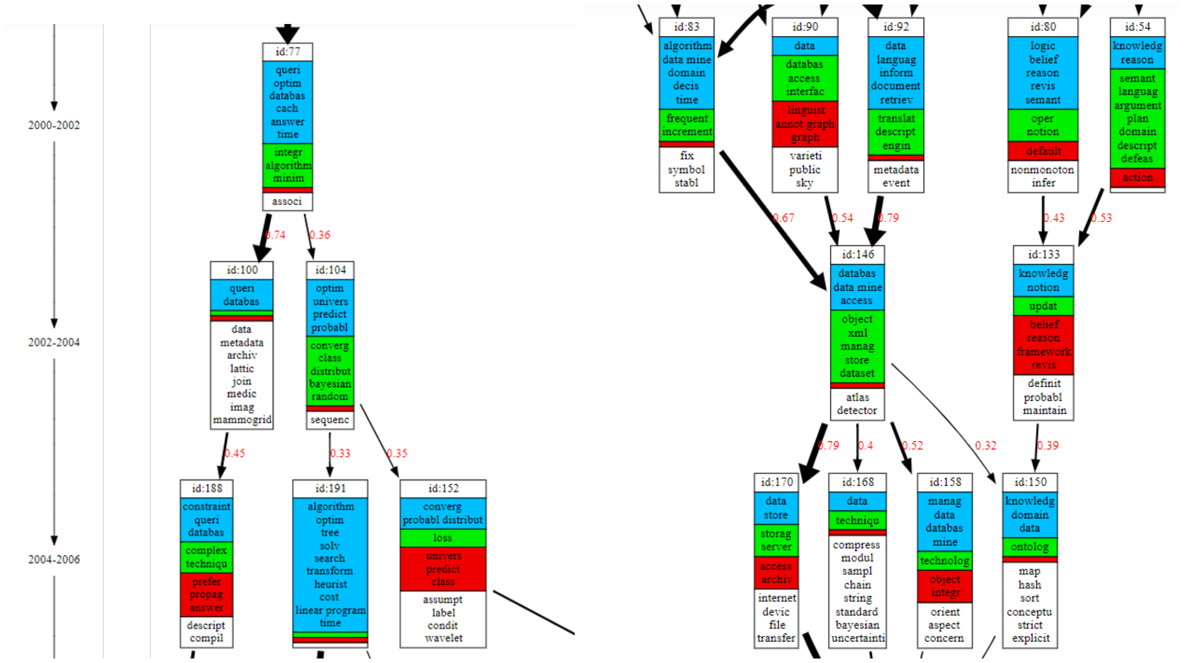


Figure 1.1: Topics containing term "database" extracted from arXiv, green = emerging terms, blue = stable terms, red = decaying terms

which exist both, in ancestor topics and in descendant topics, are grouped in blue boxes and specific terms which appear only in the current topic are in white. The thickness of the alignment edges reflects the similarity of the connected topics (e.g., cosine similarity on term vectors). Several topics in both subgraphs contain the term "database" and we can observe different evolution patterns. The left subgraph shows that in period 2002 − 2004*, topic 77 ("databases, queries, optimization, integration") splits in two research directions "databases, queries and constraints" (topics 100, 188) and "prediction, probability, random" (topics 104, 191, 152). The right subgraph covers the same period with topics related to "data mining" (83), "data access interfaces" (90), "information retrieval" (92), "logics, semantics" (80) and "knowledge, reasoning" (54). The first three topics converge in* 2002 − 2004 *into a single topic on "object, xml, store, data mining" (146) which splits in the period of* 2004 − 2006 *into "storage servers" (170), "data technique" (168), "data mining and management" (158) and "knowledge and ontologies" (150).*

1.2 The EPIQUE project

This thesis has been financed by the ANR EPIQUE project[5]. The main goals of the EPIQUE project are:

- The definition and implementation of new innovative tool for the reconstruction and exploration of multi-scale dynamics in complete real-world scientific corpora and for obtaining new insights in the evolution of complex human generated knowledge and information.
- The definition of a uniform framework for specifying, implementing and integrating large-scale text and graph mining tasks which can be customized independently of the higher-level mining algorithms with respect to specific cost models and hardware constraints (memory, CPU).
- The validation of classical hypotheses concerning the evolution of scientific fields and content and to test and improve these hypotheses in the light of the reconstructed phylomemies and of general patterns detectable within them. From the perspective of philosophy of science, EPIQUE should enable the empirical validation of theories on science evolution which have been formulated by considering only a few canonical texts. Preliminary results on small document collections covering particular scientific fields already demonstrate that phylomemetic graphs reveal novel semantic insights about science evolution [19]. Complete scientific archives like the Web of Science can more reliably be seen as a plausible testimony of scientific activity and taking into account of the whole corpus not only applies to more scientific fields but also reveals a deeper understanding of inter-disciplinary evolution.

To achieve these goals, the EPIQUE project includes four academic partners:

- Computer Science Laboratory of Sorbonne Université (Lip6, project coordinator): The LIP6 Database research team has a long experience of research in large-scale distributed data management, data integration, web data processing and data quality.
- Institute for History and Philosophy of Sciences and Techniques (IHPST): IHPST is the leading French laboratory in philosophy of science and has produced works on epistemology of computer simulation and the epistemology of big data science that are relevant to the current project.

[5] https://iscpif.fr/epique/

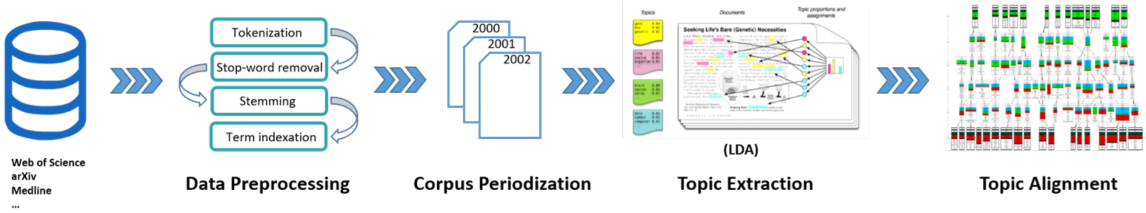


Figure 1.2: EPIQUE Workflow

- Complex Systems Institute of Paris Ile-de-France (ISC-PIF): ISC-PIF is an interdisciplinary research and training center of CNRS that promotes the development of French, European and international strategic projects on complex adaptive systems. ISC-PIF helps its partners to pool their resources for large corpora mining and analysis, and is in charge of several technological platforms in the domains of high performance computing, big data, digital humanities or visualisations.
- Research Institute of Computer Science and Random Systems (IRISA Rennes): The IRISA DRUID research team is specialized in data modeling, data protection and crowdsourcing.

This thesis mainly concerns the goal of defining and implementing the general EPIQUE framework for generating and analyzing phylomemetic networks. This framework is based on scalable text and graph mining algorithms and integrated in a workflow composed of four main steps (Figure 1.2):

1. The term extraction step applies preprocessing (term extraction, stop-word removal, deformation, term extraction) and transforms each document into a set of weighted terms.
2. The transformed documents are then grouped according to periods predefined by the user.
3. The topic detection step consists first of detecting topics (sets of strongly semantically related terms) in the term sets. In the framework of the project, we consider different ways of topic extraction, and in particular word embedding [20], word co-occurrence [21] and probabilistic models (LDA [22]).
4. Subjects extracted from different time intervals are then compared to each other, for example using the Jaccard [23] distance or the cosine distance, to generate the phylomemetic graph with alignments (split, merge, equivalent) representing their temporal evolution. The analysis and customization step of the phylomemetic graph that allows

experts to interact with the workflow by generating and visualizing phylomemetic trees and by customizing the workflow by modifying the data (e.g. removing or adding a term in a topic) and the parameters (e.g. the time interval for splitting the document collection).

Example 2 *The evolution graphs in Figure 1.1 are obtained by applying a workflow to the subsets of documents covering each time period. This workflow includes a standard NLP document preprocessing step and applies the Latent Dirichlet Allocation (LDA) method to extract a predefined number of weighted term vectors (topics) describing the scientific publication activity for each period. Finally, topics from two subsequent periods are linked by applying cosine similarity [24].*

Facing archives of increasing size, the EPIQUE workflow is not a sequence of independent tasks on a given dataset, but an integrated framework which allows experts to interact and to control the whole process through high level languages and interfaces (e.g. for specifying the scientific field and time-range of interest).

1.3 Research Challenges

This thesis addresses three main issues when building and exploring topic evolution networks:

1.3.1 Challenge 1: Scalability

Scalability is the first important challenge of our work. Most current science evolution workflows focus on domain-specific datasets. For instance, [19] studies the dynamics of phylomemies using documents in the domain of embryology. Computer Science articles from DBLP are analyzed in [25] to map the evolution of scientific fields. [26] conducts a lead-lag analysis to identify different topic evolution patterns for preprints and papers in astrophysics, etc. The size of these datasets is limited. Our first challenge is to build global maps of the evolution of science on large scientific domains by applying appropriate scientometric models on large databases like the Web of Science, MedLine or Open Archives likes arxiv. It is an ambitious goal of making sense of unstructured text through generic data processing tasks (term extraction, stop-word removal, stemming, index term generation and term selection) which become complex when dealing with very large amounts of digitized text. In this thesis we explore the development of new scalable solutions exploiting recent parallel data processing frameworks like Hadoop [27], Spark [28], and Pregel [29].

1.3.2 Challenge 2: Tuning and Quality

The second challenge concerns the quality of phylomemetic networks. The generation of topic evolution networks is a difficult task including complex unsupervised text and graph mining algorithms. Its workflow usually consists of several steps, which can be performed in different ways by choosing different algorithms and parameters. Building "meaningful" topic evolution networks is an iterative process where domain experts must choose the right algorithms and parameters for each step of the phylomemy generation workflow, especially for the topic extraction step and the topic alignment step, and correctly tune method-specific hyper-parameters and thresholds with respect to a given dataset and an expected output. Each method depends on parameters which can significantly affect the quality of topic sets and the quality of evolution graphs. Most of these parameters are dataset-specific and require empirical tuning. Besides, the tuning process also includes the threshold-based filtering of topic alignment edges to produce phylomemies for different levels of detail. To solve this challenge, we explore different solutions to accelerate the graph generation workflow and to assist experts in choosing the optimal number of topics per period to produce highly divers topic evolution networks.

1.3.3 Challenge 3: Interactive Exploration and Analysis

Figure 1.3 gives an example of a phylomemetic graph extracted from a subset of the arXiv corpus covering 20 years of publications. This corpus is split into 20 periods where each period contains 50 topics. If a corpus is extremely large and the content is quite diverse, we can obtain a very large topic evolution network like Figure 1.3 which is too complex to be visually analyzed. Existing graph visualisation standards and tools like Gephi [6] [30] or Graphviz [7] [31] can be used to generate high-quality visualisations, but their use for exploring large graphs and identifying meaningful evolution patterns is still limited. An important goal of our thesis is to allow experts to explore phylomemies interactively by filtering interesting subgraphs according to particular evolution patterns. For example, a

[6] https://gephi.org/
[7] https://www.graphviz.org/

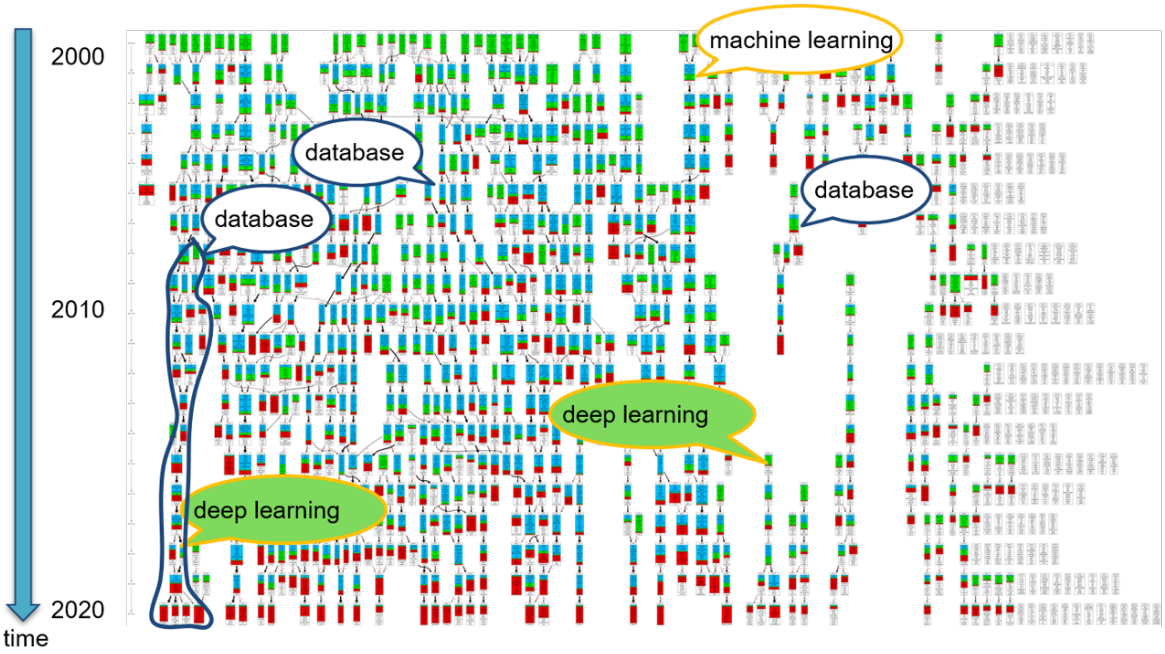


Figure 1.3: A phylomemetic graph over arXiv corpus

user may want to find all topics containing the term "database" and connected by a path to a future topic with emerging term "deep learning" in such a large and heterogeneous graph. This is practically impossible to achieve by exploring the visual representation and without any filtering capacities. Some users also want to spot interesting evolution patterns by their structure. For example, find topics of which the diameters of their patterns are more than 5 and which evolve into more than 3 domains. But evidently, most of the patterns in Figure 1.3 are connected together consisting a huge part of the phylomemy making it hard to identify important patterns. By adjusting the topic alignment thresholds, it is possible to reduce the complexity, but this also reduces the opportunity to detect interesting evolution patterns. To solve this challenge, we introduce a new query language for filtering topics and topic sub-graphs according to their content and structure.

1.4 Contributions and thesis outline

The main contributions of this thesis are listed as follows:

1.4.1 Topic Evolution Framework

Topic Evolution Model and Query Language We proposed a generic framework for the computation and interactive exploration of evolution networks. This framework includes a high-level data model using standard document and data processing technologies for extracting, storing and exploring meaningful topic evolution networks. This model relies on the notion of *pivot topic graphs* describing the content and the evolution dynamics of topics at different levels of detail, and includes a *high-level filter-based* query language which enables users to interactively explore the evolution of topics by composing structural, temporal and semantic topic filters and specify their search criteria in a simple and sound way. These topic filters consist of structural conditions on the evolution graph properties like average out- and indegree, and temporal conditions on the topic term trends like the emergence and decay of terms. This generic topic evolution framework allows users to interactively explore and analyze topic evolution networks (phylomemies). The extraction of meaningful evolution patterns from very large document archives through the framework will be introduced in Section 3.1 of Chapter 3 and the model for the interactive exploration of large topic evolution networks will be explained in Chapter 4. The presentation of the model also includes a formal analysis of the monotonicity properties of pivot filter expressions.

Diversity-based Topic Number Estimation In this model, we also defined a quality metric based on topic diversity which guarantees that a topic evolving toward many topics actually represents a significant evolution. We proposed a topic extraction method that automatically computes for each period, a set of topics that meets a given diversity condition. This diversity-based measure for estimating the quality of a topic set solves the challenge of topic number tuning and helps users to extract representative topics. The definition of this measure will be given in Section 3.2 of Chapter 3.

Prototype Implementation We implemented a scalable proof of concept prototype on top of Apache Spark for processing large scientific corpora containing millions of documents and finding meaningful topic evolution networks for both stable topics and highly evolving ones. The prototype including a pivot graph generation interface (notebook) and an interface (notebook) for exploring and analyzing pivot graphs is able to address all previous challenges and will be illustrated in Chapter 6.

1.4.2 Pivot graph computation

Two Topic Alignment Strategies To generate multi-stage topic evolution networks, we proposed two alignment strategies where one takes account of the similarities of all topic pairs which has been integrated in our prototype and another one focuses on aligning topics only from consecutive periods and uses a *nearest-neighbor* condition to select candidate topics. Both strategies addressed the problem of computing a very large number of cosine-based topic alignments on top of Apache Spark. We also highlighted that Spark's native distributed cosine similarity computation can be improved for the nearest-neighbor strategy in our context of graph evolution of scientific documents. The more detailed description will be provided in Section 3.3 of Chapter 3.

Two Pivot Graph Computation Strategies To achieve a high level of interactivity, we firstly proposed an incremental join-based transitive closure algorithm for the materialization of pivot graphs and the graph properties by computing and storing all possible aggregated values in advance. This kind of materialization is computation and storage-intensive, but also can directly benefit of standard big data technologies to achieve scalability. Its main benefit is that even complex structural and temporal topic filters can be implemented by simple value-based selections on the generated topic properties. To avoid many join operations of the join-based materialization strategy, we proposed an alternative pivot graph materialization strategy based on GraphX [32] using Bulk Synchronous Parallel Paradigm [29]. The objective of proposing these two pivot graph computation strategies is to solve the scalability challenge. These two strategies will be discussed in Sections 5.1 and 5.2 of Chapter 5.

1.4.3 Experimental Evaluation

We preprocessed several corpora of different scales, such as a small corpus about economics, Glyphosate, arXiv, Nature, Wiley and Elsevier, etc., where most of them are issued from the ISTEX [33] platform which is an online access to more than 23 million articles from all scientific disciplines. We then conducted a detailed experimental evaluation on the performance of different EPIQUE workflow steps including the LDA generation, the topic alignment, the pivot graph computation, the topic labeling and the graph metrics computation over four of the previously mentioned real-world datasets. This evaluation also includes experiments measuring the scalability of our pivot graph generation algorithm over larger synthetic topic evolution networks. These experiments prove that each step of our framework is capable of efficiently handling large datasets (scientific document collections or evolution graphs). The detailed descriptions of these experiments are distributed across the thesis and will be given in Sections 3.2 and 3.3 of Chapter 3 and Section 5.3 of Chapter 5.

1.5 List of Publications

The related publications of this thesis are listed as follows:

- (2019) **Ke Li**, Bernd Amann, Hubert Naacke. EPIQUE: Extracting Meaningful Science Evolution Patterns. Poster@BDA 2019.
- (2019) Hubert Naacke, **Ke Li**, Bernd Amann, Olivier Curé. Efficient similarity-based alignment of temporally-situated graph nodes with Apache Spark. IEEE International Conference on Big Data (Big Data).
- (2020) **Ke Li**, Hubert Naacke, Bernd Amann. EPIQUE: Extracting Meaningful Science Evolution Patterns from Large Document Archives. Demo@International Conference on Extending Database Technology (EDBT 2020).
- (2020) **Ke Li**, Hubert Naacke, Bernd Amann. Exploring the Evolution of Science with Pivot Topic Graphs. International Workshop on Big Data Visual Exploration and Analytics BigVis (EDBT 2020).
- (2020) **Ke Li**, Hubert Naacke, Bernd Amann. EPIQUE: Extracting Meaningful Science Evolution Patterns from Large Document Archives. Demo@BDA 2020.
- (2020) **Ke Li**, Hubert Naacke, Bernd Amann. EPIQUE: A Graph Data Model and Query Language for Exploring the Evolution of Science. BDA 2020.
- (2021) **Ke Li**, Hubert Naacke, Bernd Amann. An Analytic Graph Data Model and Query Language for Exploring the Evolution of Science. Special Issue on Interactive Big Data Visualization and Analytics of the Big Data Research Journal, Elsevier (under revision).

2 Related Work

Contents

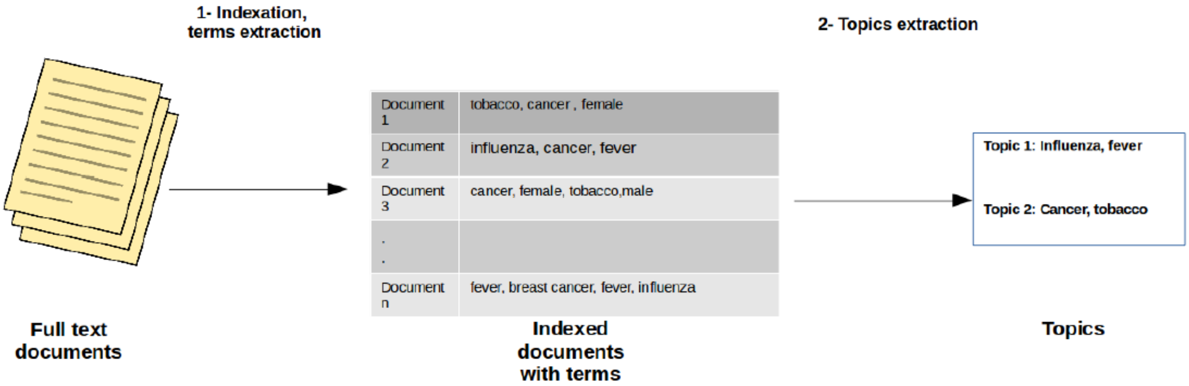
2.1 Topic Modeling

Topic modeling can be applied to solve various problems like document classification [34, 35, 36, 37], sentiment analysis [38, 39], topic discovery [22, 40] and image object localization [41, 42]. In our work, we apply topic model mainly for extracting a representative set of topics describing a collection of documents.

Most topic models are based on the assumption that groups of words describing a semantic concept (topic) will often occur together in semantically similar documents. Various methods have been applied to the topic modeling problem, such as Frequent Itemset mining [43], Community Detection and Clique Percolation [44], Word Embedding [20] and Statistical Topic Modeling [45]. All these models are based on a preprocessing phase where each text document is first transformed into a structured (sequence, graph) or unstructured (set) of terms. These structures are then analyzed taking account of the term co-occurrence in these structures. We will first give a short description of the different topic modeling approaches and then concentrate on the statistical topic model which we applied in our work.

2.1.1 Topic Modeling Approaches

- Frequent Itemset [43]: A topic is a set of terms that are relevant to one or several documents and describe their contents. One hypothesis about topic extraction is to consider the set of terms that show up together in certain number of documents as a topic. Frequent Item-set Mining consists in detecting sets of items (terms) that often appear together in the same transaction (document). It has been applied to market basket analysis where it aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies, on-line shops etc. Figure 2.1 illustrates the use of Frequent Itemset for topic extraction. Documents can be considered as transactions, while indexed terms of each document can be regarded as items. Then, from the example corpus, two topics can be extracted where one topic contains “influenza” and “fever”, and another one contains “cancer” and “tobacco”.



Document 1	tobacco, cancer , female
Document 2	influenza, cancer, fever
Document 3	cancer, female, tobacco,male
. .	
Document n	fever, breast cancer, fever, influenza

Figure 2.1: Topic extraction by Frequent Item-set Mining

- Clique Percolation [44]: The Clique Percolation Method (CPM) is a popular approach for analyzing the overlapping community structure of networks. The term network community is usually defined as a group of nodes that are more densely connected to each other than to other nodes in the network. The clique percolation topic model considers terms as nodes and topics as communities in the network. One advantage of this approach is that clusters can overlap and a given term can thus be present with various meanings in different topics.

[46] applies k-clique percolation for extracting overlapping topics (paradigmatic fields) from scientific documents and defining an asymmetric topic proximity metric to represent the hierarchical structure of scientific activity.

[47] proposes ACPM, an extension of CPM to analyze the dynamics between topics and to identify topic clusters exhibiting an increase of collaborations that potentially will lead to the emergence of new topics. The approach generates semantic enhanced topic networks selecting all keywords from the publications in a given year that also appear as concepts in the CSO[1] Computer Science Ontology [48].

- Word Embedding [20]: Word Embedding represents words as vectors (embedding) such that words with a similar meaning are represented by similar vectors. The real-valued vector representation is learned from a corpus for a predefined fixed sized vocabulary. Vocabulary terms are mapped into a high-dimensional vector space such that terms frequently used in the same context appear close together in this space. Then the identified denser regions in this space can be considered as scientific fields.

  Our EPIQUE project partners from IRISA Rennes study the structures of scientific topics and their evolution by using the Word Embedding [49]. They trained a word embedding model on the Wiley corpus of a given time period and applied the cosine similarity to compute the distance matrix of the data points. By applying a new hierarchical clustering method [49], they obtained a rich structured representation of the topic clusters extracted for different periods and their evolution.

- Statistical Topic Models [45]: Statistical topic modeling is based on the distributional hypothesis that words that are close in meaning will occur in similar documents. The analysis of the relationships between a set of documents and a set of terms (document-term matrix) produces a fixed number of topics, the distribution of the topics over the documents (document-topic matrix) and the distribution of the terms over the topics (topic-term matrix). The goal of topic modeling is to uncover these distributions (latent variables) that shape the meaning of the document and the whole corpus.

[1] http://cso.kmi.open.ac.uk/

2.1.2 Latent Topic Models

Statistical topic models have become the standard model for analyzing the content and efficiently extracting key information contained in large-scale corpus. Our topic evolution workflow also uses the statistical topic model LDA (Latent Dirichlet Alloction) which we will study in more detail in the following section.

Latent Semantic Indexing LSI [50] learns latent topics by performing a matrix decomposition on the document-term matrix using Singular Value Decomposition (SVD) [51]. LSI can be considered as a dimension reduction [52] or noise reduction [53] technique. It assumes that words (terms) that are close in their meaning will often occur together in similar documents. A matrix containing word weights (e.g. word counts) per document (rows represent each document and columns represent unique words) is constructed from a document collection and SVD is used to reduce the number of columns while preserving the similarity structure among rows.

Consider a set of m text documents using a total number of n unique terms (words). We wish to extract k topics from all the text data in the documents (k has to be specified by the user). Let A be an $n \times m$ matrix having TF-IDF [54] scores whose rows represent documents and columns represent terms. As displayed in Equation (2.1), SVD decomposes a matrix into three other matrices, matrix U, matrix S, and V^T (transpose of matrix V).

$$A = USV^T \tag{2.1}$$

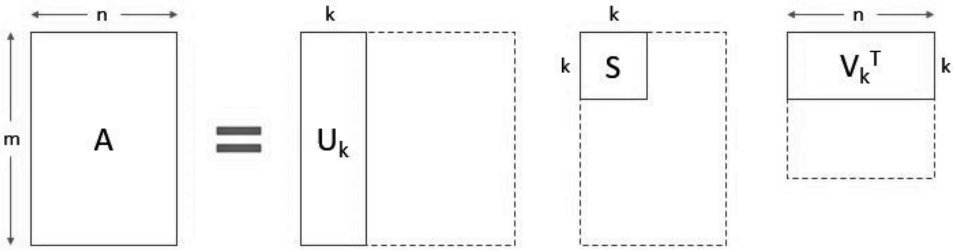


Figure 2.2: Graphical model of LSI

Figure 2.2 shows the graphical model of LSI. Each row i of the matrix U_k is a vector of length k (number of topics) which contains the weight of each topic in the corresponding document. U_k is called the document-topic matrix. The term vector representation of the k topics can be found in the matrix ${V_k}^T$ where each vector of length n (number of terms) contains the weight of each term in the corresponding topic (topic-term matrix).

The document-topic matrix U_k and topic-term matrix ${V_k}^T$ are a compact vector representation of the documents and topics which can be used to identify similar topics (term vectors) and similar documents (topic vectors) using for example *Cosine similarity* [24].

LSI can be implemented very easily. However, LSI assumes a Gaussian distribution of the terms in the documents and does not reflect most real term-document distribution, which are in general skewed. The obtained results are also difficult to interpret. Moreover, LSI involves SVD, which is computationally intensive and hard to update as new data comes up. Another challenge of SVD is that it is hard to find an optimal number of topics k.

Probabilistic Latent Semantic Analysis Instead of using matrices (SVD), probabilistic Latent Semantic Analysis (pLSA) [55] uses a probabilistic method where the core idea is to find a probabilistic model that generates documents as mixtures of a low-dimensional set of topics.

LSA models the probability of each (word, document) co-occurrence as a mixture of conditionally independent multinomial distributions as shown in Equation (2.2) with Z being the words' topic.

$$P(W, D) = P(D) \sum_{Z} P(Z|D) P(W|Z) \tag{2.2}$$

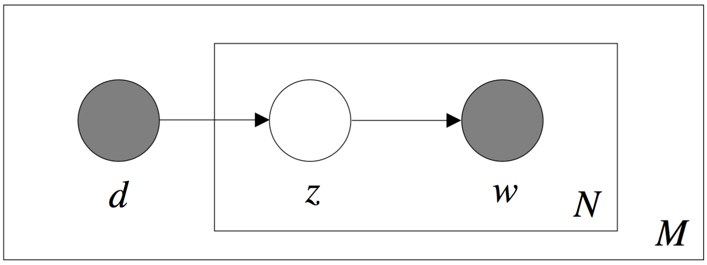


Figure 2.3: Graphical model of pLSA

Figure 2.3 shows the graphical model of pLSA as a Bayesian network where the boxes are "plates" representing replicates. The outer plate M represents documents, while the inner plate N represents the repeated choice of topics and words within a document. Each node is a variable in the model and edges encode probabilistic dependencies between the two variables. Shaded nodes denote that the variables were observed. Given a document d (with probability $P(d)$), a topic z is present in that document with probability $P(z|d)$ and given a topic z (with probability $P(z)$), a word w is drawn from z with probability $P(w|z)$.

The results of pLSA have a clear probabilistic interpretation. However, the number of parameters grows linearly with the number of documents and it is difficult to assign probabilities to documents which are not part of the training set.

Latent Dirichlet Allocation LDA [22] is a Bayesian version of pLSA and follows the intuition that the probability distribution over words is skewed. It therefore applies a sparse Dirichlet prior [56] to model the per-document topic and per-topic word distributions.

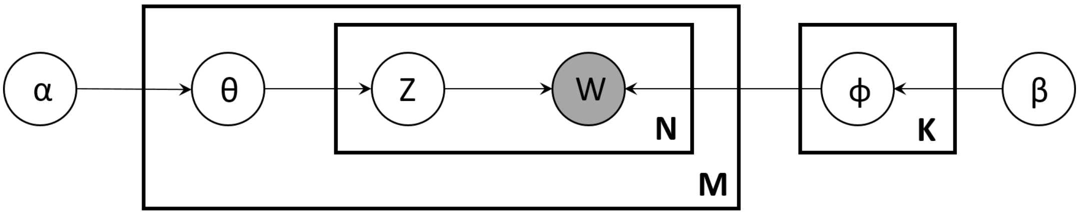


Figure 2.4: Graphical model of LDA

Figure 2.4 shows the graphical model for LDA. From a dirichlet distribution $Dir(\alpha)$, LDA draws a random sample representing the topic distribution, or topic mixture, of a particular document. This topic distribution is θ. From θ, it selects a particular topic Z based on the distribution. Next, from another dirichlet distribution $Dir(\beta)$, LDA selects a random sample representing the word distribution of the topic Z. This word distribution is ϕ. From ϕ, the particular word W is finally chosen.

Formally, the process for generating each word from a document is as follows:

1. Choose random per document topic distribution vector $\theta_i \sim Dir(\alpha)$ (where $i = 1,...,M; \theta_i \in \Delta_K$)
   - $\theta_{i,k}$ = probability that document $i \in 1,...,M$ has topic $k \in 1,...,K$.
2. Choose random per topic word distribution vector $\phi_k \sim Dir(\beta)$ (where $k = 1,...,K; \phi_k \in \Delta_V$)

- $\phi_{k,v}$ = probability of word $v \in 1, ..., V$ in topic $k \in 1, ..., K$.

3. Choose topic from the multinomial distribution θ_i $z_{i,j} \sim Multinomial(\theta_i)$ (where $z_{i,j} \in 1, ..., K$)

   - $z_{i,j}$ = is a topic

4. Choose word from the multinomial distribution $\phi_{z_{i,j}}$ $w_{i,j} \sim Multinomial(\phi_{z_{i,j}})$ (where $w_{i,j} \in 1, ..., V$)

   - $w_{i,j}$ = is the weight of word j in topic $z_{i,j}$

LDA extracts human-interpretable topics from a corpus, where each topic is characterized by the words it is most strongly associated with. LDA can generalize to new documents easily and only needs the definition of two Dirichlet priors (word and topic distribution) and an additional parameter K which denotes the number of topics to be generated. We have addressed the problem of choosing the optimal number of topics by proposing a diversity-based measure (Section 3.2.1).

Hierarchical Dirichlet Process HDP [57] is an extension of LDA which addresses the case where the number of topics is not known a priori. Much like LDA, HDP models topics as mixtures of words of a certain number of topics. However, the number of topics is not a predefined constant, but a random variable generated by a Dirichlet process. A common base distribution is selected which represents the countably-infinite set of possible topics for the corpus, and the finite distribution of topics for each document is sampled from this base distribution.

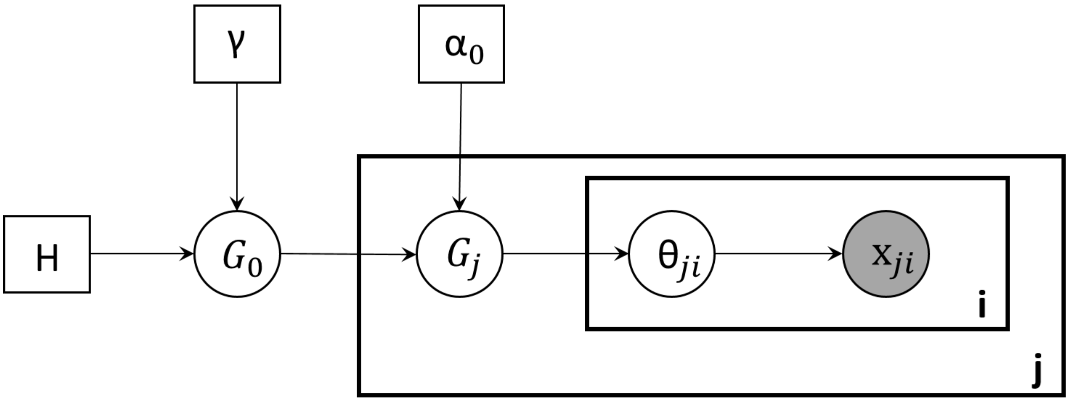


Figure 2.5: Graphical model of HDP

HDP is a nonparametric topic model which allows the mixture models (documents) to share components (topics). More precisely, a base distribution draws topics (term-distributions) from an uncountable set of topics. Then each document would be constructed from a completely unique set of topics. By feeding the base distribution into a Dirichlet process, the topics are allowed to be shared across documents. Thus, it is a distribution over a set of random probability measures: one probability measure G_j which is a distribution over a countably-infinite number of topics for each document j, and a global probability measure G_0 which is a distribution over a countably-infinite number of topics. As shown in Figure 2.5, the global probability measure G_0 is distributed as $DP(\gamma, H)$, with H the base measure and γ the concentration parameter, such as

$$G_0|\gamma, H \sim DP(\gamma, H) \tag{2.3}$$

The probability measure G_j can be obtained from the upper level of DP sampling, such as

$$G_j|\alpha, G_0 \sim DP(\alpha_0, G_0) \tag{2.4}$$

If an HDP model can be used as a grouped data about the prior term distribution of θ_{ji}, for any document j, it assumes that $\theta_{j0}, \theta_{j1}, ..., \theta_{ji}$ are independent identically distributed random topics of G_j, such as

$$\theta_{ji}|G_j \sim G_j \tag{2.5}$$

Each term distribution θ_{ji} can be used to generate the corresponding observed term x_{ji}, such as

$$x_{ji}|\theta_{ji} \sim F(\theta_{ji}) \tag{2.6}$$

.

With HDP, the maximum number of topics can be unbounded and learned from the data rather than specified in advance. However, it has been shown that HDP is inconsistent for estimating the number of topics, which means that the posterior distribution on the number of clusters does not converge to the true number of topics, even with an infinite amount of data [58]. Furthermore, HDP does not scale to very large document corpus.

2.1.3 Dynamic Topic Modeling

The goal of dynamic topic models [59, 60, 61, 62] is to capture the evolution of topics in a sequential document corpus. These models not only extract topics from documents for different time periods, but also detect trends of the term usage within these topics. This allows them to achieve better accuracy than static topic models for the prediction of the topics of a given period from the topics of the previous period.

For example, [60] develops a Dynamic Topic Model (DTM) where corpus are divided into different time slices using the document timestamps. The topics of each time slice are modeled using LDA where the topics associated with one slice evolve from topics associated with the last slice. For taking account of the temporal dimension during the topic extraction process, DTM chains together the topic parameters at each time slice using a linear Kalman filter [63]. Each topic is drawn using a logistic normal distribution [64] and defines an evolution path from the topic of the first slice to the topic of the last slice.

Topics over Time (TOT) [62] is different from DTM. Instead of modeling topic evolution using discretized time periods or the Markov assumption [65] over state transitions, TOT parametrizes a continuous distribution over time where topics generate both timestamps as well as words. Parameter estimation is thus driven to discover topics that simultaneously capture word co-occurrences and localization of those patterns in time. Figure 2.6 gives the graphical model of TOT. As can be seen, TOT is a generative model of timestamps and the words in the timestamped documents, and all the timestamps of the words in a document are observed as the same as the timestamp of the document.

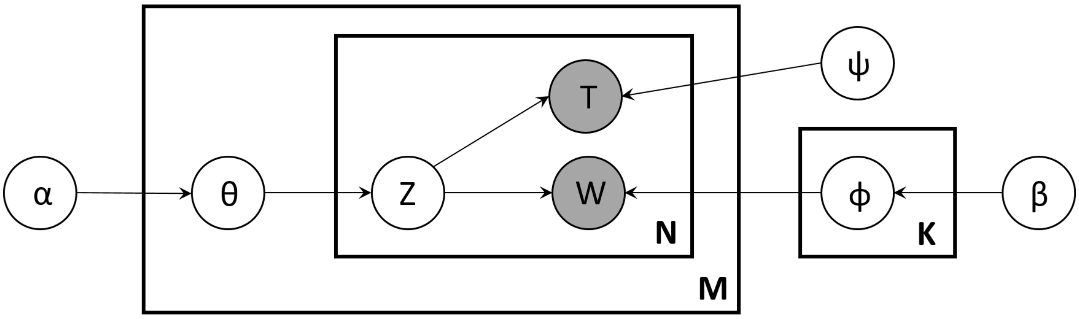


Figure 2.6: Graphical model of TOT

[59] introduces a novel inference algorithm which makes fewer restricting assumptions by using Gibbs Sampling [66]. They also present a Metropolis-Hastings [67] based sampler for

topic assignments for each word token. Their algorithm achieves lower perplexity [68] and is also orders of magnitude faster than the baselines in both single machine and distributed environments.

In our work, we are interested in generating and exploring topic networks connecting similar topics from different time periods and the use of such dynamic models would obviously have sense. However the computation complexity quickly increases as time granularity increases and we decided to apply a different solution. In order to achieve a "smooth term semantics shift" in topics from different periods, we define overlapping time periods to extract subsets of documents and apply static LDA topic extraction to each overlapping subset. As our experiments show, this strategy allows us to produces meaningful topic alignments with lower processing costs. The comparison with an approach using dynamic topic models over disjoint periods is an open future work.

2.2 Topic Evolution Models

Since we only focus on the change of evolution patterns extracted from the textual document content in our work, we will review some literature on the cognitive view of topic evolution in this section. This kind of topic evolution models mainly can be distinguished between topic trend analysis and topic evolution networks.

2.2.1 Topic Trend Analysis

Topic detection and trend analysis studies the temporal evolution of topics within a document stream. The process consists of detecting emerging topics and following their evolution including their decay. We can distinguish between two topic trend analysis tasks: (1) the *topic detection* task detects new topics in the incoming document stream, generally by applying unsupervised clustering methods and (2) the *topic tracking* task which, given a seed of documents and their topics, monitor the document stream for further documents on the same topics.

[69] is the first article which gives a full conversion of an unsupervised Topic Detection system into a supervised Topic Tracking system. They use an unsupervised cluster-dependent algorithm for the topic detection as discussed in [70] which assigns each document a cluster

by computing the document-cluster similarity. Then if a document has been labeled as a seed document of a topic, any subsequent document that is placed into the same cluster of the seed document is a candidate for labeling as on-topic for the topic of the cluster. Topic trend analysis has been applied to follow event-based topics across incoming document streams, such as newswire documents [69], US Patent database [71], etc., and to detect emerging trends in scientific literature [72].

Topic Trend Analysis in Scientific Literature

A Gibbs Sampling based implementation of LDA has been applied by [73] to analyse 28 154 abstracts published in PNAS from 1991 to 2001. The authors proposed a method for estimating the optimal number of topics (based on the log likelihood) by using Bayesian model selection [74] and studies the evolution of topics by applying a linear trend analysis on the mean θ values (document distributions over the topics) by year.

[75] also applies unsupervised topic modeling to the ACL Anthology[2] to analyze historical trends in the field of Computational Linguistics from 1978 to 2006 by using the observed probability of each topic given the current year. They induce topic clusters using LDA,and use the trends in these topics over time and over conference venues to address questions about the development of the field. They found that three conferences are converging in the topics they cover by applying Jensen-Shannon divergence [76] of topic distributions.

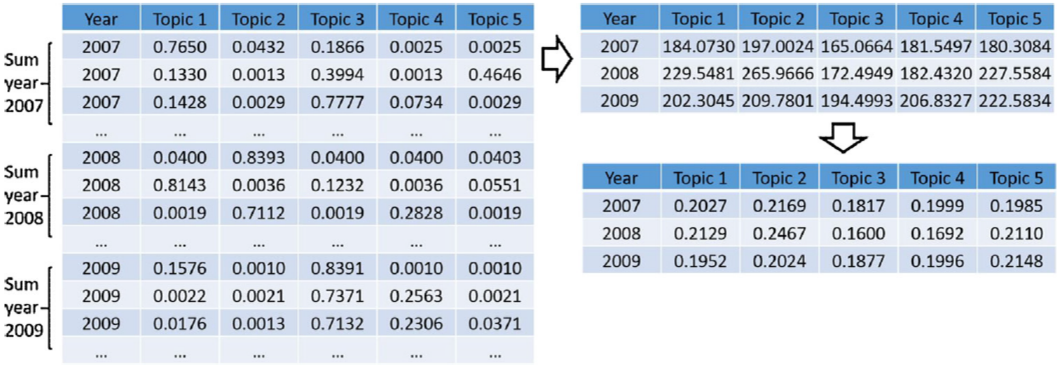
Hu et al. [26] is another example which applied LDA and regression analysis to identify different topic evolution (topic popularity and duration of topic popularity) patterns for preprints and papers from arXiv and the Web of Science (WoS) in astrophysics for the last 20 years $(1992 - 2011)$. The number of publications in WoS and the number of preprints in arXiv along the years for each topic have been modeled as curves by a regression model. The paper redefines the notion of topic trend and popularity, and demonstrates that topics in WoS lose their popularity much earlier than similar topics in arXiv and open access preprints (like arXiv) have stronger growth tendency as compared to printed publications.

How “cognitive science” as a field has changed over the last three decades has been explored by [77] using LDA topic model. They proposed a topical weighted-contribution method (weighted contribution of each topic by year) to analyze trends in scientific fields over time, focusing on the 34 years of Cognition articles published between 1980 and 2014

[2] https://www.aclweb.org/anthology/

(3104 abstracts). They have found that over the last three decades, Cognition articles are increasingly framed in terms of experiments, rather than abstract theories.

Breakthrough research may not be the mainstream area, but can attract a significant amount of citations. For that reason, [78] develops a graphical model of Topical Impact over Time (TIoT) to capture the temporal dynamics in the impact of latent topics from a corpus of documents. They propose a LDA-style topic model using citation counts to quantify topical impact, which can be used for detecting trending topics and suggesting impactful papers in a bibliographical database. This model can be used for the design of digital libraries and social media platforms, as well as evaluation of scientific contributions and policies.



	Year	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Sum year-2007	2007	0.7650	0.0432	0.1866	0.0025	0.0025
	2007	0.1330	0.0013	0.3994	0.0013	0.4646
	2007	0.1428	0.0029	0.7777	0.0734	0.0029
	...	...	...	...	...	...
Sum year-2008	2008	0.0400	0.8393	0.0400	0.0400	0.0403
	2008	0.8143	0.0036	0.1232	0.0036	0.0551
	2008	0.0019	0.7112	0.0019	0.2828	0.0019
	...	...	...	...	...	...
Sum year-2009	2009	0.1576	0.0010	0.8391	0.0010	0.0010
	2009	0.0022	0.0021	0.7371	0.2563	0.0021
	2009	0.0176	0.0013	0.7132	0.2306	0.0371
	...	...	...	...	...	...

Year	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
2007	184.0730	197.0024	165.0664	181.5497	180.3084
2008	229.5481	265.9666	172.4949	182.4320	227.5584
2009	202.3045	209.7801	194.4993	206.8327	222.5834

Year	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
2007	0.2027	0.2169	0.1817	0.1999	0.1985
2008	0.2129	0.2467	0.1600	0.1692	0.2110
2009	0.1952	0.2024	0.1877	0.1996	0.2148

Figure 2.7: Steps for generating topic popularity introduced by [79]

Topic trend analysis can be summarized as the steps shown in Figure 2.7. Topic models like LDA are applied to the entire corpus. Then a document-topic matrix can be obtained as the result which contains for each document the topic distribution. This distribution can be considered as the topic contribution to the corpus. Since each document has a timestamp, topic trends like topic popularity can be acquired by aggregating the topic contribution by time period.

2.2.2 Topic Evolution Networks

Topic trend analysis explores the temporal evolution of different topic properties like popularity, importance, utility or interest. Whereas the set of analyzed topics can evolve in time, trend analysis does not take account of the topic relationships and the structural evolution of topics in time. Our work models the evolution of topics by networks connecting similar topics from different time periods without measuring and comparing their distribution, popularity and impact.

Topic evolution networks represent the structural evolution of topics in time, considering that during some time period topics can split into two or more new more sub-topics or be merged with other topics into a single new topic. These structural changes are generally represented by directed acyclic networks connecting (aligning) topics extracted from documents published at different periods. Existing evolution network based frameworks mainly can be distinguished by the chosen topic extraction (Section 2.1) and topic alignment methods.

Topic Alignment Measures

The first two methods are term-based similarity measures which are frequently used to compute text similarity:

- *Cosine similarity* [24] calculates similarity by measuring the cosine of angle between two vectors. With cosine similarity, we need to convert topics into vectors of term distribution. Here is the cosine similarity formula:

  $$similarity = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

  It will generate a metric that says how related are two vectors by looking at the angle. The Cosine similarity of two vectors will range from 0 to 1. If the Cosine similarity value is 1, it means two vectors have the same orientation. The value closer to 0 indicates that the two vectors have less similarity.

  The cosine similarity performs well on distinguishing the correlations between sparse vectors, where it highlights the contribution of the top rank terms with high probabilities and weaken the noise produced by the terms with low probabilities. In our work, LDA topics are sparse term vectors where each vector only contains no more than 10 most important terms, while all other terms have much smaller weights. This alignment approach is also used in the work of [80].

- *Jaccard similarity* [23] measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

  $$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Here $\boldsymbol{A}$ and $\boldsymbol{B}$ are topics of term sets where the appearance of each term indeed makes sense instead of the frequency. The Jaccard similarity value is in a range of 0 to 1. If the two term sets are identical, Jaccard similarity is 1. The Jaccard similarity value is 0 if there are no common terms between two topics. [19, 80] applied this measure to align scientific topics in their works.

Here are some differences between Cosine similarity and Jaccard similarity:

1. Cosine similarity takes total length of term vectors while Jaccard similarity takes only unique set of the terms for each topic.
2. Cosine similarity is good for cases where duplication matters while analyzing topic similarity whereas Jaccard similarity is good for cases where duplication does not matter.

The following functions measure the similarity or distance between two discrete probability distributions:

- *Bhattacharyya coefficient (BHD)* [81] is a measure of similarity between two probability distributions. Here we use p and q to represent term vectors, p_i and q_i to represent the weights of the i-th term and n_v to represent the length of vectors:

  $$B(p,q) = \sum_{i=1}^{n_v} \sqrt{p_i q_i}$$

  It is the maximum value of 1 which is attained when there is the greatest degree of similarity between p and q (i.e. $p = q$) and 0 the least. [82] and [80] modeled the evolution of topics using the Bhattacharyya coefficient.

- *Hellinger distance* [83] is another measure to quantify the similarity between two probability distributions. With the same parameters as mentioned above, we have:

  $$H(p,q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n_v} (\sqrt{p_i} - \sqrt{q_i})^2}$$

  Also,

  $$1 - H^2(p,q) = \sum_{i=1}^{n_v} (\sqrt{p_i q_i})$$

  Therefore, It is straightforward to demonstrate that the Hellinger distance is related to

Bhattacharyya coefficient $B(p,q)$ as it can be defined as

$$H(p,q) = \sqrt{1 - B(p,q)}$$

It takes on a value between 0 and 1, with 0 signifying the greatest degree of similarity between p and q (in this case $p = q$) and 1 the least. The work of [80] tracked the topical structural changes by applying the Hellinger distance as well as BHD and Jaccard similarity.

- *Kullback-Leibler divergence (KLD)* [84] (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution. It is a distribution-wise asymmetric measure. For example, the Kullback-Leibler divergence from q to p is defined to be

  $$D_{KL}(p||q) = \sum_{i=1}^{n_v} p_i log(\frac{p_i}{q_i})$$

  A Kullback-Leibler divergence of 0 indicates that the two distributions in question are identical, otherwise it can take values between 0 and ∞. It measures the amount of information lost in the approximation of the probability distribution p_i with q_i. KL divergence is not a real distance metric because it is not symmetric. For example:

  $$D_{KL}(p||q)! = D_{KL}(q||p)$$

  [82] found that BHD and KLD can capture different types of topic relatedness. BHD can be used to track gradual topic evolution, speciation, and convergence, whereas KLD can used to detect topic splitting and merging.

- *Jensen-Shannon divergence* [76] is known as another method of measuring the similarity between two probability distributions. It is defined as the average of the KL divergence of each distribution to the average of the two distributions:

  $$JSD(p||q) = \frac{1}{2}D_{KL}(p||m) + \frac{1}{2}D_{KL}(q||m)$$

  where

  $$m = \frac{1}{2}(p+q)$$

  It is a symmetric, smoothed and normalized version of the KL divergence with values between 0 (identical) and 1 (maximally different), and can be used as a distance metric.

Combined with cosine similarity, Jensen-Shannon divergence is employed in [85] to compare with their proposed topic similarity algorithm.

Topic Evolution Network in Scientific Literature

There exist fewer works of topic evolution network than the works of topic trend analysis. Most of them concentrate on the evolution of scientific fields by analyzing scientific literature.

[19] comes up with a method to enable a bottom-up reconstruction of the dynamics of scientific fields and they apply it to two large-scale case studies: "embryology research" and "networks in biology" corpora. They generate topics using directed cliques of co-occurring terms and align inter-temporal scientific fields by Jaccard similarity [23] in their phylomemetic networks. Their work demonstrates that their bottom-up approach is well-adapted to revealing the robust patterns of science evolution.

[82] is the first work that discusses and distinguishes between two groups of particularly challenging topic evolution phenomena: topic splitting and speciation and topic convergence and merging. They apply Hierarchical Dirichlet Process (HDP) [57] to generate topics and Bhattacharyya similarity [81] to track gradual topic evolution, speciation, and convergence. The alignment process also applies (asymmetric) Kullback-Leibler divergence (KLD) [84] for detecting topic split and merge.

Using a data set on information retrieval (IR) publications, [79] examines how research topics evolve by analyzing the topic trends from documents and evolving dynamics which presents the splitting and merging of topics and the underlying knowledge transfer among topics. For this study, they apply LDA to extract global topics and use the aggregation of the per-document topic distribution by year as their popularities to detect topic trends. From each time span, they extract local topics using LDA. The correlation between local topics and between a local topic and a global topic is measured by cosine similarity [24]. Then the splitting and merging of local topics indicates the existence of knowledge transfer within a global topic or between global topics.

[80] proposes a HDP-based framework for the discovery of the topical content of a data corpus and the tracking of its complex structural changes across the temporal dimension by using Hellinger distance [83], BHD [81] and Jaccard similarity [23]. In constructing a similarity graph they use a threshold to eliminate automatically weak edges, retaining only

the connections between sufficiently similar topics in adjacent epochs. Instead of using a similarity threshold, they prune the graph based on the the operating point on the cumulative distribution function (CDF) [86]. They found that the introduction of some overlap (25-50% of the epoch length) in successive epochs could significantly enhance the relatedness of topic nodes in an evolution graph.

Our thesis adapts the notion of phylomemy from the work [19] of one of our partners, ISC-PIF, and extends previous works on topic evolution networks by proposing a formal query language based on a set of semantic and structural graph filters to define complex topic evolution patterns and a query-by-example interface to visualize topic evolution graphs at different levels of detail.

Table 2.1 shows a summary of the presented approaches for modeling the evolution of topics in scientific archives.

Table 2.1: Topic Models for Trend and Evolution Analysis

Ref	Topic model	Tuning	Alignment method	Evolution	Use case
[73]	LDA on entire corpus	log likelihood using bayesian model selection	trend analysis on the document-topic distribution by year	trend	28154 abstracts in PNAS (1991-2001)
[75]	LDA on entire corpus	manual	empirical probability p(z∣y) that an arbitrary paper d written in year y was about topic z	trend	historical trends in the field of Computational Linguistics of ACL Anthology from 1978 to 2006
[26]	LDA on entire corpus		regression to fit trend curve of number of publications for each topic	trend	arXiv and the Web of Science (WoS) in astrophysics (1992-2011)
[77]	LDA on entire corpus	topics that have a significant topic contribution	weighted contribution of each topic per year (proportion of words generated by each topic)	trend	Cognition articles published between 1980 and 2014 (3104 abstracts)
[78]	LDA-style topic model on entire corpus		model citation counts as topic impact	trend	D-Lib Magazine and The Library Quarterly from 2007 to 2017
[19]	directed clique of co-occuring terms	co-occurrence proximity measure	Jaccard distance	network	WOS (embryology research 200 000 articles from 1991 to 2010) and Medline (networks in biology 140 000 articles)
[82]	HDP on each period	non-parametric	BHD to track gradual topic evolution, speciation, and convergence KLD to detect topic splitting and merging	network	Pubmed (22,508 articles on ASD (autism spectrum disorder) and 31,706 on MetS (metabolic syndrome))
[79]	LDA on entire corpus for global topics LDA on each period for local topics	empirical analysis	topic trend: per-document topic distribution by year topic evolution: Cosine similarity	trend network	WoS (20 359 documents in Information retrieval for 1956 - 2014)
[80]	HDP on each period	non-parametric	Hellinger distance, BHD, Jaccard distance	network	Pubmed (22,508 articles on ASD (autism spectrum disorder) and 31,706 on MetS (metabolic syndrome))

3 Phylomemy Generation

Producing meaningful topic evolution graphs is difficult and needs an important tuning effort and expertise in statistical text mining. In the EPIQUE project, we aim at building a generic topic alignment workflow for the extraction of meaningful evolution patterns from very large document archives and the interactive exploration of large topic evolution graphs.

This chapter precisely explains the phylomemy generation workflow (Section 3.1) as well as a diversity-based quality measure for the extraction of representative topics (Section 3.2.2). Finally, it describes two efficient topic alignment strategies which are capable of computing large cosine-based topic alignments on top of Apache Spark(Section 3.3).

Contents

3.1 Phylomemy Generation Workflow

Figure 1.2 illustrates the main steps of the EPIQUE phylomemy generation workflow. The workflow mainly consists of 4 steps transforming a document archive into a phylomemy: (1) Data preprocessing (tokenization, stopword removal, stemming, term indexation), (2) Corpus periodization, (3) Topic extraction and (4) Topic evolution alignment. Each step produces intermediate results for the next step and the quality of the final result obviously depends on these intermediate results. We will describe each step in detail and explain its influence on the final result.

Workflow Input The input of the EPIQUE workflow can be any scientific document corpus where each document has at least a publication date and a text describing the content of the publication such as the title, the abstract or the keywords. All scientific archives like Web of Science[1], arXiv[2], Medline[3], DBLP[4], etc. satisfying these simple conditions can be exploited by the workflow. Observe that we do not take account of any other metadata like the authors, venue (journal, conference) or bibliographic references (see Chapter 1 on Page 2). The quality of the final result depends on the textual input but also some other parameters which will be described later. As we will see in Section 3.2.2, in particular the diversity of the extracted topics depends on the variety and the amount of available relevant terms describing the scientific content of each document.

3.1.1 Data Preprocessing

The workflow starts with a standard text preprocessing pipeline of lexical analysis, stopword removal, stemming, index term generation and term selection. The main goal of this step is to extract for each document a weighted list of terms (term vector) which precisely describe its content.

Our implementation of this step allows experts to choose among two term extraction methods where the first method indexes terms by using a predefined domain-specific vocabulary list and the second method extracts important words by applying an automatic term extraction

[1] https://clarivate.com/webofsciencegroup/solutions/web-of-science/
[2] https://arxiv.org/
[3] https://www.nlm.nih.gov/medline/medline_overview.html
[4] https://dblp.org/

process implemented using the Stanford natural language processing toolkit, CoreNLP [87] which provides multiple *Computational Linguistics* [88] tools. More precisely, we have applied the *depparse()* function which allows to extract a graphical representation, *i.e. dependency tree* [89], of a sentence that provides a simple description of the grammatical relationships between words in a sentence. For example, the *amod* relations contain all adjectival modifiers with their modified noun phrases. Combined with part-of speech tags [90] generated by the *pos()* function, we can easily spot adjectives and adverbs, and then remove them as one of our textual data cleansing steps. The list of extracted keywords obviously plays an important role for the following topic extraction step which has a direct impact on the quality of the final phylomemetic graph and might need some additional tuning by providing domain-specific vocabularies or adding new terms to the stop term list.

3.1.2 Corpus Periodization

This step consists of decomposing the term vectors obtained for input document according to several contiguous, possibly overlapping, time windows. Each time window defines a *corpus period*, which is the subset of documents published during the corresponding time period. Windows might overlap which means that the same document may appear in two successive periods. The choice of the window size and the overlap degree (sliding=overlap, landmark=no overlap) is significant for several reasons. The window size controls the number of documents in each period which is important for the following topic extraction step which requires a minimal number of documents as input. The overlap degree controls the "semantic topic shift" between two periods which decreases with increasing overlap. The window size depends on the granularity of the document time-stamps (year, month, day) and the number of available documents in each period. We assume that these two parameters are fixed by the expert depending on the input corpus and the kind of evolution study she wants to undertake. As we will see in Section 3.2.2, an important goal in the choice of these parameters is to achieve for each period a highly diverse set of topics.

3.1.3 Topic Extraction

In this step, each corpus period is fed to a topic model for extracting a set of topics that can be aligned in the following topic alignment step. The output of the topic model describes each topic as a set of terms or a weighted term vector. The choice of the output (term sets or weighted term vectors) depends on the alignment method used for aligning topics from different periods.

We use the Latent-Dirichlet-Allocation method (LDA) [22] implemented by the machine learning library Spark MLlib [91] to extract topics from each corpus period. LDA produces a topic-term matrix and a document-topic matrix. The former allows us to construct topic evolution networks (phylomemies) and the latter allows for providing additional analysis such as topic importance, topic-document distribution, etc. Among the different input parameters (see Chapter 2 on Page 18), LDA also requires a fixed number of topics defined by the user. This parameter influences the diversity of the extracted topics and the quality of the final phylomemy networks. A low topic number generates a small set of generic topics which are insufficient to distinguish specific scientific domains whereas a high number might generate many similar topics as it will be explained in Section 3.2.2

3.1.4 Topic Evolution Alignment

A phylomemetic network represents the evolution links connecting topics from different periods. The weight of a link between a topic t from a period to a topic t' of the next period expresses the probability that t has evolved into t'. This probability is estimated by the similarity between t and t' and called the evolution degree. We can define different networks by applying different similarity thresholds. These thresholds determine the complexity of the evolution graph and the evolution degree of topics. Higher thresholds create simpler phylomemies with more strongly connected topics and less alignment connections whereas lower thresholds allow to generate more complex phylomemies connecting many dissimilar topics.

Topics generated by LDA are weighted term vectors representing the term distribution in topics. These vectors are aligned with an appropriate similarity measure. It is possible to choose among different similarity measures, like Jaccard distance [23], Cosine similarity [24], Bhattacharyya distance (BHD) [81] or Hellinger distance [83] to estimate the evolution similarity between two topics in two different time periods. In our implementation, we applied cosine similarity because the term distribution is a sparse vector and cosine similarity performs well on measuring the correlations between sparse vectors. Furthermore, Apache Spark proposes a distributed map-reduce [92] implementation of cosine similarity, which is efficient for our high-dimensional weighted vector setting.

The quality of a phylomemy mainly depends on the quality of the topics extracted from the previous step. Besides this quality issue, the performance to compute a phylomemy has to be taken into consideration as well. We will show in the following section how our system assists experts in efficiently producing high quality phylomemies.

3.2 Topic Extraction

LDA is widely used for various different tasks and implemented in many ML libraries, such as Apache Spark MLlib [91], Scikit-Learn[5] et Gensim[6]. Its popularity can be explained by its high performance and simple usage since it only requires the definition of some basic parameters like the number of topics to be extracted and the size of the vocabulary to be used for indexing. The quality of the topic model is generally evaluated using indicators like *Perplexity* [68] which is a measurement of how well a probability model predicts a sample and *Likelihood* [93] which measures the goodness of fit of a statistical model to a sample of data for given values of the unknown parameters. However, our experiments have shown that these two measures are not well adapted to our goal of generating many diverse topics covering different scientific domains. Moreover, [94] has shown that perplexity and likelihood are not strongly correlated to human judgment because they do not consider the context and semantic associations between words. In this section, we will introduce a diversity-based measure for estimating the quality of a topic set and show some experimental results for different scientific archives.

3.2.1 Diversity Estimation

In order to build phylomemies over more representative topic sets, we propose *topic diversity* for estimating the quality of a topic set. The topic diversity inside a period can be estimated by observing the dissimilarity distribution over all topic pairs inside the period.

Suppose that T represents a set of topics over a set of periods $\mathcal{P}$. We denote by $p_i \in \mathcal{P}$ the period of topic t_i and two topics t_i and t_j are in the same period if $p_i = p_j$. We denote by $T_k = \{t_i | t_i \in T \wedge p_i = p_k\}$ the set of topics at time period $p_k \in \mathcal{P}$. Observe that time periods can overlap in time, but each topic is attached to a single period. Finally, we denote by T_p

[5] https://scikit-learn.org/stable/

[6] https://radimrehurek.com/gensim/

a set of topics in the same period $p \in \mathcal{P}$ and $sim : T \times T \rightarrow [0,1]$ a topic evolution function estimating the *evolution similarity* between any topics in T. We use cosine similarity as the *sim* function which will be explained in detail in Section 3.3. Then the diversity ratio of topics in T_p is computed by the following formula:

$$D_{T_p} = \frac{|(1 - sim(t_i, t_j)) > s|}{|(t_i, t_j)|} \text{ where } t_i, t_j \in T_p \tag{3.1}$$

In the above formula, s is a dissimilarity threshold which allows to filter highly divers topic pairs. Observe that $D_{T_p} \in [0,1]$. A high value signifies that most topics in T_p are dissimilar, whereas a low value indicates that there are many similar topics in T_p.

Experimental results using this diversity measure are shown in Section 3.2.2.

3.2.2 Experimental Evaluation

We defined two experimental evaluations concerning the topic generation step. The first series of experiments evaluate the scalability of the parallel Apache Spark's LDA implementation. The second series of experiments illustrate the importance of choosing the right number of topics for achieving high diversity.

These experiments were conducted on four real-world data sets of different scales by using the titles and the abstracts of each document. The smallest dataset contains 4640 documents about research related to the Glyphosate herbicide. The second dataset ISTEX contains 13 423 articles in the domain of ecological economics and environmental economics. The arXiv corpus is a repository of electronic preprints approved for publication after moderation. This repository consists of 1.15 million scientific publications in the fields of mathematics, physics, astronomy, electrical engineering, computer science (arXiv.CS), etc. The last dataset is a sample of the Wiley online library which contains 1 million documents covering the fields of arXiv and additional fields such as agriculture, art, humanities, etc.

The statistics for these four data sets are summarized in Table 3.1, where *#Document* is the total number of documents and *#T* is the number of topics per period we used in our experiments which is chosen according to our proposed diversity-based quality measure (Section 3.2.2). All datasets cover 20 years of publications which are split into 10 periods by using a sliding 3-year time window with an overlap of 1 year. The number of documents covered by each period is different but remains in the same order of magnitude. We did not apply any sampling before generating the document-term matrix and executing LDA topic extraction.

Table 3.1: Dataset statistics

Datasets	*#Document*	Period	*#Periods* (total)	$\#T$/period
Glyphosate	4640	1994 – 2013	10	30
ISTEX	13 423	1991 – 2010	10	30
arXiv	1 156 114	1998 – 2017	10	100
Wiley	1 023 515	1996 – 2015	10	200

The computation is executed on top of Apache Spark version 2.4, Scala version 2.11 and Java version 8. Spark allows parallelization at the CPU core level and at the machine level (cluster). Our experiments have shown that most steps (except the data preprocessing step) can be efficiently executed on a single machine where the performance depends on the number of CPU cores. In the following experiments on the real world datasets, we show the results obtained by the execution on a single machine with a hyperthreaded 3.1 GHz Intel Core i7-7920HQ processor (4 CPU cores), 16 GB RAM and a 512 GB SSD disc.

LDA Computation and Scalability

The goal of the first experiment is to evaluate the scalability of the parallel Apache Spark's LDA implementation. *LDA* is applied on a document-term matrix containing the term frequency for each term/document pair. This matrix is obtained by preprocessing the full archives of raw text documents (title + abstract). The preprocessing includes special character removal, tokenization, stopword removal, stemming and term generation (including frequent ngram detection). This preprocessing takes up to 6 hours for large archives like Wiley ($>$ 1M documents), but is completely automatic and has to be done only once.

Figure 3.1 displays the *LDA* performance on a single machine. We evaluated the total *LDA* execution time on the document term matrix of each corpus *wrt.* different CPU core numbers. For the two small corpus Glyphosate and ISTEX, *LDA* takes about $1-2$ minutes. It is interesting to see that the cost slightly increases with more CPUs, which can be explained by an increase of the parallelization overhead. For the larger corpus with a million documents such as arXiv and Wiley, each LDA model takes between 2.5 and 7.5 minutes and the parallelization overhead is compensated by the benefit of parallelizing the LDA tasks.

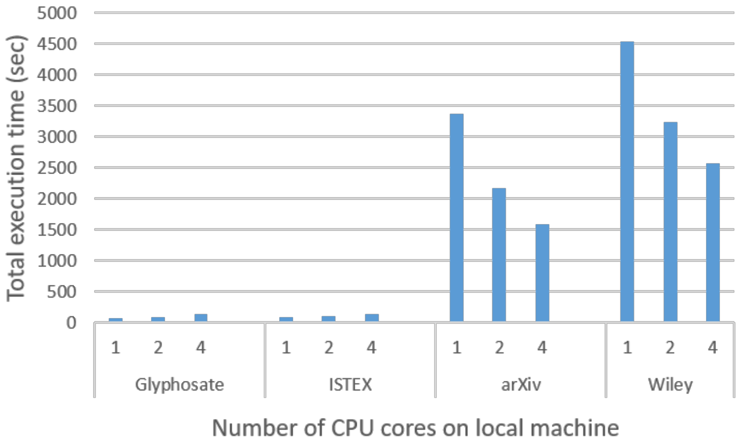


Figure 3.1: LDA performance on local machine *wrt*. number of CPU cores

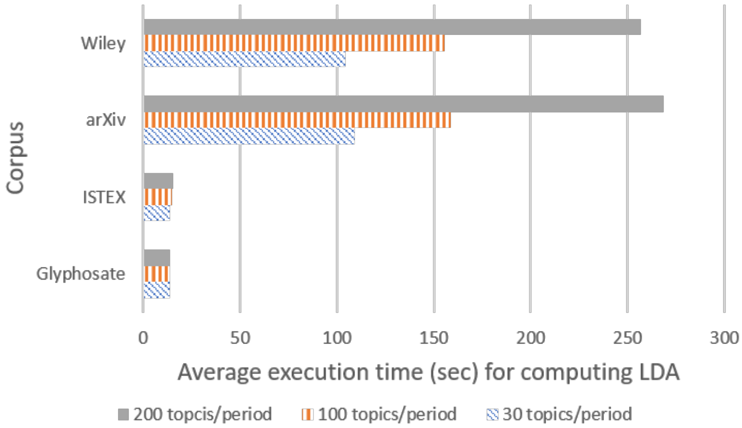
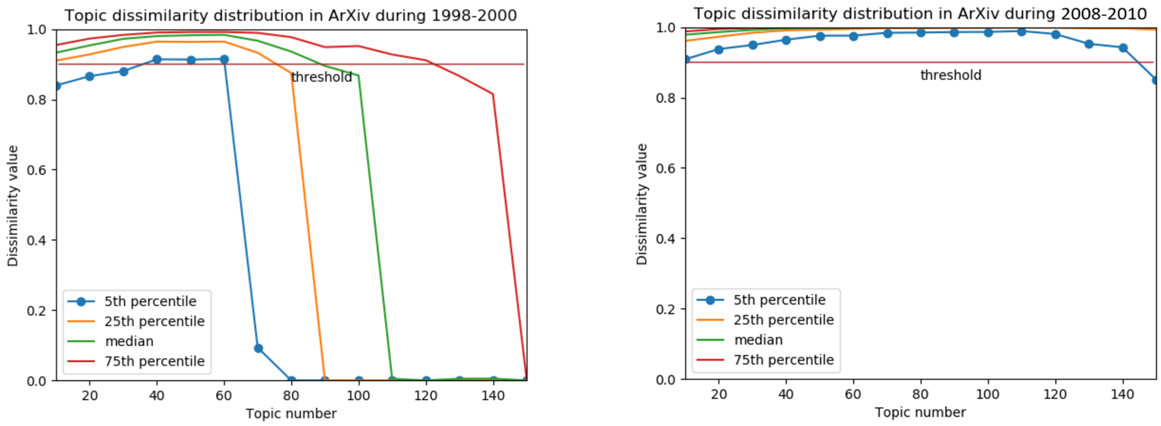


Figure 3.2: Average execution time for computing LDA VS. #*T*

Figure 3.2 presents the the average execution time for computing the LDA model per period by using 4 CPU cores with respect to different numbers of topics (#T) for each corpus. The performance of LDA computation mainly depends on the number of documents and the number of extracted topics per period. For example, for Glyphosate with 4640 documents and 30 topics per period, LDA takes about 15 seconds, whereas it almost takes ten times more for the same number of topics and about 1 million documents (arXiv). For the arXiv corpus, it takes about 110 seconds to compute a LDA model with 30 topics, whereas extracting 200 topics doubles the execution time.

Diversity Computation and Results



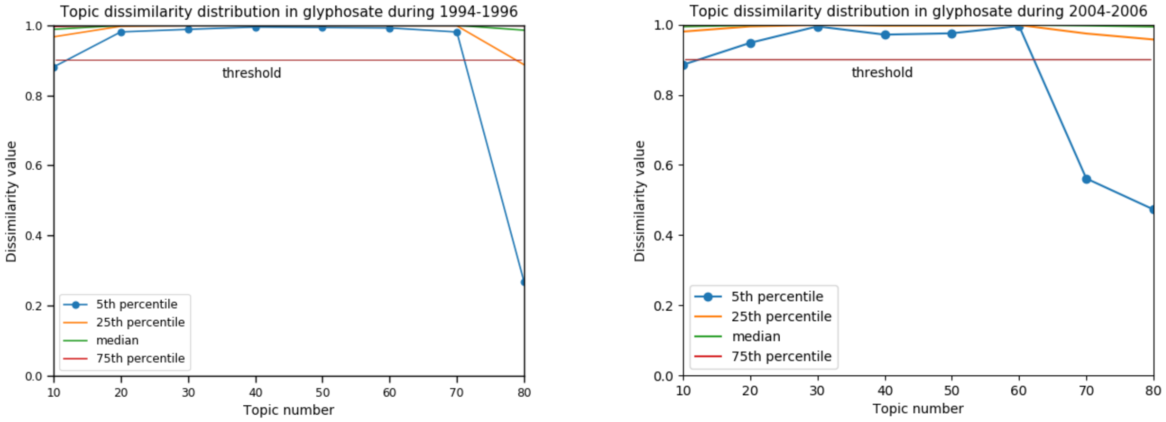
(a) 1998-2000

(b) 2008-2010

Figure 3.3: Dissimilarity distribution (diversity) by number of topics in arXiv.CS

The goal of this experiment is to show that our topic diversity measure allows to find a range of optimal topic number #T for a given corpus and the number of optimal topic numbers depends on the size of the corpus. Figure 3.3a and Figure 3.3b show the topic diversity obtained for different LDA models applied to 1164 documents published in arXiv.CS during 1998 to 2000 and 16 072 documents published in arXiv.CS during 2008 to 2010. Each LDA model corresponds to a different number of topics #T ranging from 10 to 150. For example, for the smaller corpus, we can see in Figure 3.3a that for a topic number #T ranging between 40 and 60, less than 5 percent (blue line) of all topic pairs have a dissimilarity value lower than 0.9 and any number in this range is a good choice. Figure 3.3b shows that for the larger corpus, LDA achieves high diversity even for 140 topics.

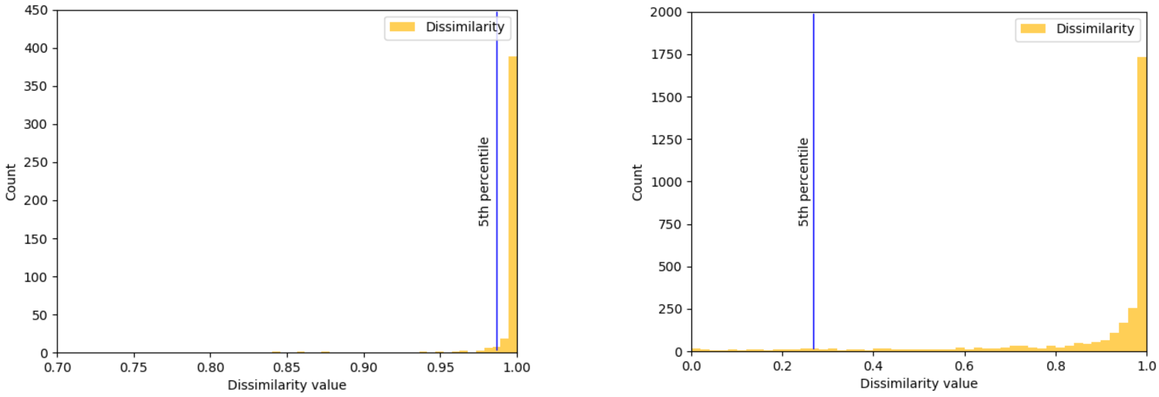
Figure 3.4a and Figure 3.4b give two other examples of topic diversity evaluation over the



(a) 1994-1996

(b) 2004-2006

Figure 3.4: Dissimilarity distribution (diversity) by number of topics in Glyphosate



(a) Dissimilarity distribution for 30 topics

(b) Dissimilarity distribution for 80 topics

Figure 3.5: Dissimilarity distribution of topic pairs in Glyphosate during 1994-1996

Glyphosate corpus obtained by applying different LDA models to 489 documents published during 1994 to 1996 and 679 documents published during 2004 to 2006. It can be easily seen that the similar amount of documents leads to similar behaviors in terms of the range of optimal topic number #*T* estimated by the topic diversity measure.

Figure 3.5a and Figure 3.5b display the detailed dissimilarity distribution for 30 topics and 80 topics extracted from the documents in Glyphosate corpus published during 1994 to 1996, which correspond to the dissimilarity distribution for 30 topics and 80 topics in Figure 3.4a. In Figure 3.5a, more than 95% of all topic pairs have a dissimilarity value higher than 0.985 (where the 5*th* percentile line is located) which indicates a pretty good topic diversity value since our diversity threshold is 0.9. Whereas Figure 3.5b is an example of topics with low diversity where the 5*th* percentile value of the dissimilarity distribution only reaches 0.27, which reveals more similar topics in the topic set.

Our experiments on the different scientific document archives have confirmed that the diversity of topics produced by LDA mainly depends on the size of the analyzed document set (see Table 3.1). In our implementation, we propose experts to generate the visual diversity diagram on a chosen period and to choose a fixed number of topics in the optimal range, where higher topic numbers produce topics described by a more specific vocabulary than lower numbers. The extension to an automatic grid-search based topic number estimation step is straightforward.

3.3 Topic Alignment

Phylomemies might contain thousands of topics and alignment edges which are computed by comparing millions of topic pairs with some similarity function. In this section, we are addressing the efficiency of topic alignment computation on top of the unified analytical Apache Spark engine.

To generate multi-stage topic evolution graphs, we propose two topic alignment strategies where one takes account of the similarities of all topic pairs which is important for our topic evolution model (Chapter 4), whereas another one focuses on aligning topics only from consecutive periods and uses a nearest-neighbor condition to select candidate topics. For the second strategy, we highlight that Spark's native distributed cosine similarity computation can be improved in our context of graph evolution of scientific documents.

3.3.1 Full Matrix Alignment Computation

The formal alignment model is based on graphs for representing the evolution of a set of topics T over a set of periods $\mathcal{P}$. We use the same notions as defined in Section 3.2.1. The Spark's distributed cosine similarity implementation takes as input a single set of topics T and computes the set of all similarity values between topics in T:

$$S_T = \{sim(t_i, t_j) | t_i, t_j \in T\} \text{ where } sim(t_i, t_j) = \frac{t_i.t_j}{||t_i||.||t_j||} \tag{3.2}$$

This is a *MapReduce* algorithm which is a programming model and an associated implementation introduced by Google [92] for processing and generating big data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a *map* procedure, which performs filtering and sorting (such as mapping words in a document by the number of occurrences, the value 1 for each word), and a *reduce* method, which performs a summary operation (such as counting the sum of the number of occurrences for each word, yielding word frequencies).

This MapReduce algorithm consists in decomposing each topic vector $t_i \in T$ into n sub-vectors (partitions) and distribute these sub-vectors t_i^k, $1 \leq k \leq n$ across all the cluster nodes. Considering t_i as a vector of length d (terms), and using a cluster of M nodes, then each topic is partitioned into M sub-vectors of length $q = d/M$ and the sub-vectors of all topics at the same position k are stored on the same node N_k. Figure 3.6 illustrates the data structure of a topic-term matrix implemented by RowMatrix[7], which is a row-oriented distributed matrix on Spark. In the matrix, each column corresponds to a topic vector of length d, and is partionned into M sub-vectors of length d/M and distributed across all the cluster nodes. A first map step locally computes on each node N_k the square of the partial norms of every t_i^k:

$$||t_i^k||^2 = \sum_{(k-1)q < l \leq kq} t_i[l]^2$$

The following reduce step sums the partial norms to get the final norms replicated on all nodes:

$$||t_i|| = \sqrt{\sum_{1 \leq k \leq n} ||t_i^k||^2} = \sqrt{\sum_{1 \leq l \leq d} t_i[l]^2}$$

[7] https://spark.apache.org/docs/2.2.0/api/java/org/apache/spark/mllib/linalg/distributed/RowMatrix.html

A second map step evaluates the partial scalar products of every sub-vector couple (t_i^k, t_j^k):

$$t_i^k.t_j^k = \sum_{(k-1)q<l\leq kq} t_i[l].t_j[l]$$

And a final reduce step computes the complete scalar products and the final cosine similarity values on all nodes:

$$t_i.t_j = \sum_{1\leq k<n} t_i^k.t_j^k$$

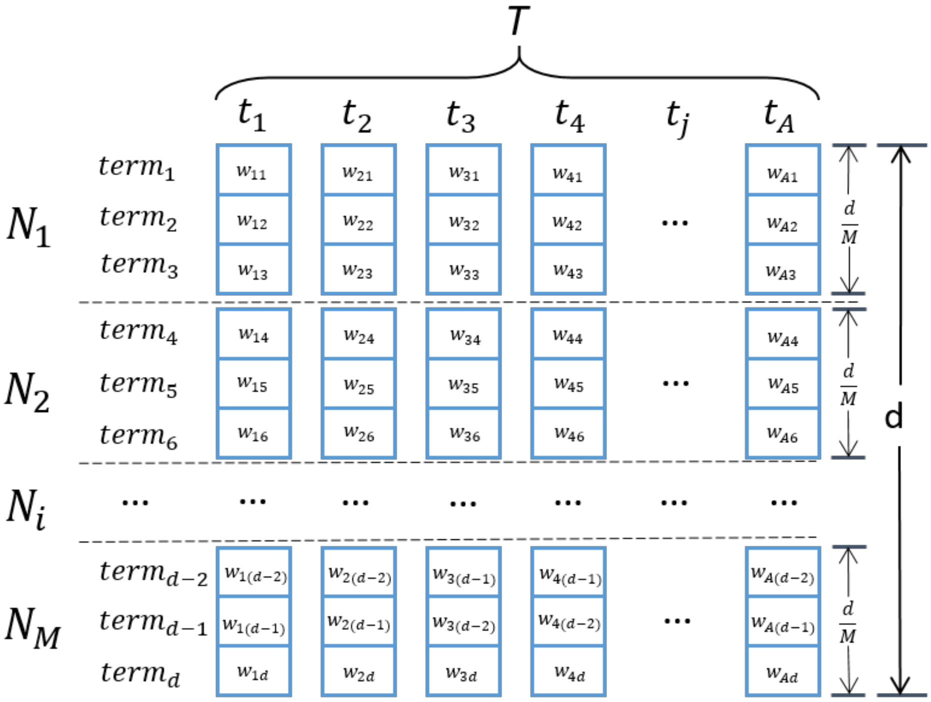

T
t_1 t_2 t_3 t_4 t_j t_A
N_1
$term_1$ w_{11} w_{21} w_{31} w_{41} w_{A1}
$term_2$ w_{12} w_{22} w_{32} w_{42} ... w_{A2}
$term_3$ w_{13} w_{23} w_{33} w_{43} w_{A3}
$\frac{d}{M}$
N_2
$term_4$ w_{14} w_{24} w_{34} w_{44} w_{A4}
$term_5$ w_{15} w_{25} w_{35} w_{45} ... w_{A5}
$term_6$ w_{16} w_{26} w_{36} w_{46} w_{A6}
$\frac{d}{M}$
d
N_i ... ...
N_M
$term_{d-2}$ $w_{1(d-2)}$ $w_{2(d-2)}$ $w_{3(d-1)}$ $w_{4(d-1)}$ $w_{A(d-2)}$
$term_{d-1}$ $w_{1(d-1)}$ $w_{2(d-1)}$ $w_{3(d-2)}$ $w_{4(d-2)}$ ... $w_{A(d-1)}$
$term_d$ w_{1d} w_{2d} w_{3d} w_{4d} w_{Ad}
$\frac{d}{M}$

Figure 3.6: Distributed matrix implementation in Spark

This algorithm is implemented by the function *columnSimilarities()* of class RowMatrix. The *columnSimilarities()* function computes locally on each node cosine similarities between any pair of sub-vector columns at the same position and finally returns an $n \times n$ sparse upper-triangular matrix of cosine similarities between columns of this matrix as shown in Figure 3.7.

Since our topic evolution model which will be presented in Chapter 4 takes account of the evolution (similarity) between topics from distant periods, we propose a *full-matrix* alignment computation strategy in the following. This alignment computation strategy adopts the idea

$$\begin{array}{c|ccccccc|}
 & t_1 & t_2 & t_3 & . & . & . & t_A \\
t_1 & 0 & s_{12} & s_{13} & . & . & . & s_{1A} \\
t_2 & 0 & 0 & s_{23} & & & & s_{2A} \\
t_3 & 0 & 0 & 0 & & & & . \\
. & . & & & . & & & . \\
. & . & & & & . & & . \\
. & . & & & & & . & . \\
t_A & 0 & . & . & . & . & 0 & 0
\end{array}$$

Figure 3.7: Upper-triangular matrix of cosine similarities between topic vectors

that sub-vectors of the *complete* topic-term matrix (containing all topic vectors from all periods) are distributed across all cluster nodes and the partial evolution similarities of all topic pairs are computed on each node. The similarities between topics of the same period are also calculated to evaluate the topic diversity of each topic period.

In our implementation, the largest topic set is from the Wiley corpus (among the real-world datasets that we use) containing 2000 topics in total (200 topics per period). This number of topics can easily be managed by current big data framework on a single node.

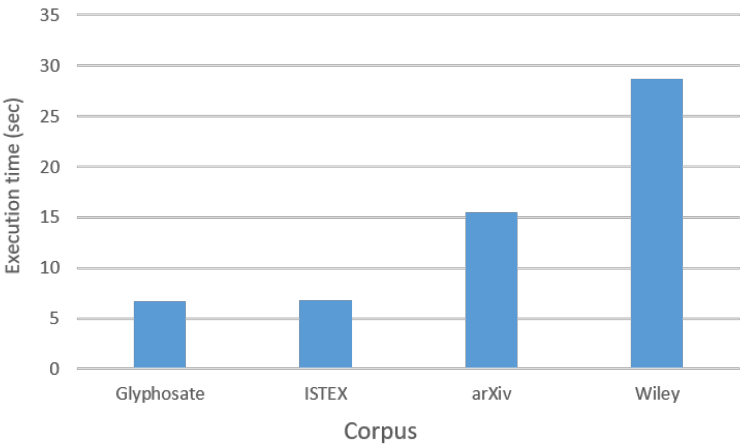


Figure 3.8: Full matrix alignment computation performance on local machine

Figure 3.8 shows the performance evaluation of full matrix alignment computation for the 4 datasets on a local machine with the settings as described in Section 3.2.2. The number of topics of each corpus is reported in Table 3.1. As can be seen, the execution time grows linearly with the number of topics. More precisely, 300 topics are extracted from the the Glyphosate corpus and the ISTEX corpus respectively, and it takes about the same time to compute the full matrix for both corpora. The total number of topics extracted from the arXiv corpus is 1000 and the alignment computation takes three times larger than the two previous datasets. Then for the Wiley corpus (2000 topics), the full matrix computation time is 2 times longer than the the execution time for arXiv.

3.3.2 Nearest-Neighbor Alignment Computation

The aforementioned strategy computes similarities of all topic pairs which results in a computation of a full topic-term matrix. But we are also interested in computing multi-stage topic evolution graphs by aligning topics only from consecutive periods. Thus, we consider a nearest-neighbor alignment approach that keeps an edge connecting t_1 with t_2 if and only if:

1. $t_1 \in T_1$, $t_2 \in T_2$, with $p_2 = p_1 + 1$
2. $sim(t_1, t_2) > sim(t_1,$ nearest neighbor of t_1 in $T_1)$
3. $\forall t_i \in T_1, sim(t_1, t_2) \geq sim(t_i, t_2)$

In other words, t_2 gets aligned with t_1, if t_2 from period $p_2 = p_1 + 1$ is closer to t_1 than any other topic in p_1, and if t_1 is the best candidate among all other topics of p_1.

The overall number of similarity values to compute is $P \times N_{sim1}(A) + (P-1) \times N_{sim2}(A)$ with:

- T_i set of topics in period p_i
- $A = |T_i|$ topics per period[8]
- P periods
- $N_{sim1}(A) = \frac{A \times (A-1)}{2}$ similarity values per period for finding the similarity of the nearest neighbor of each topic in the same period (alignment condition 2)
- $N_{sim2}(A) = A^2$ values for two consecutive periods (alignment link candidates)

In the following paragraphs, we propose two different implementation approaches to compute the previously defined nearest-neighbor topic alignments which are both based

[8] For simplification we assume that all periods contain the same number of topics. This is coherent with the use of LDA which allows to fix the number of topics generated for each period.

on the cosine similarity between two topic vectors t_i and t_j. Both approaches mainly differ on their ability to efficiently distribute topic sets based on their periods over a cluster of computers. As shown in the experiment of Section 3.3.2, this distribution ability drastically impacts the performance of these approaches.

Period-ordering Alignment Computation

In the nearest-neighbor alignment model, only consecutive topic periods are aligned. Formally given topic sets T_1, T_2 and T_3 occurring at three consecutive periods p_1, p_2 and p_3, only the alignments from topics in T_1 to topics in T_2 ($T_1 \rightarrow T_2$) and the alignments from topics in T_2 to topics in T_3 ($T_2 \rightarrow T_3$) need to be computed. Observe that in this alignment approach, the evolution from topic sets T_1 to T_3 is "transitively" captured through alignments with T_2. We denote by $U_i = T_i \cup T_{i+1}$ for $1 \leq i \leq P-1$ the topics at consecutive periods p_i and p_{i+1}. By definition, $|U_i| = 2 \times A$. This allows us to rely on Spark's distributed cosine similarity method for computing S_{U_i}, *i.e.*, the similarity between every pair of topics in U_i. By iterating over the sequence $U_1, \cdots, U_{m-1}$, we get the evolution similarities S_{U_i} for all periods and select the expected alignments that satisfy the alignment conditions as defined before (items 2 and 3).

In the general case, each node computes the partial norms and scalar products for all possible pairs of topics which are then combined in two shuffle/reduce steps. We estimate the data shuffling cost generated by this distributed computation. We can consider that the shuffling cost per computed similarity value is constant and therefore proportional to the number of computed values. Thus, the shuffle cost of finding the nearest neighbor of each topic in a period can be represented as follows:

$$C_{SHUFFLE} = N_{sim1}(A) \times u \times M = \frac{A \times (A-1)}{2} \times u \times M \tag{3.3}$$

where A is the number of topics in T, u is the (constant) shuffle cost (number of bytes) per similarity value and M is the number of computation nodes.

Then the alignment cost for aligning all topics from two consecutive periods is displayed as follows:

$$\begin{aligned} C_{ALIGN} =& (P-1) \times C_{U_i} = (P-1) \times N_{sim1}(|U_i|) \times u \times M && (3.4)\\ =& (P-1) \times \frac{|U_i| \times (|U_i|-1)}{2} \times u \times M && (3.5)\\ =& (P-1) \times A \times (2 \times A - 1) \times u \times M && (3.6) \end{aligned}$$

A first observation is that this implementation computes all similarities between two topics of the same period p_i twice, once for the alignment $T_{i-1} \rightarrow T_i$ pair and once again for the $T_i \rightarrow T_{i+1}$ pair. For P periods, the amount of redundant similarity values is $(P-2) \times \frac{A \times (A-1)}{2}$. Secondly, it also misses opportunities to parallelize the alignment of period pairs, *e.g.*, aligning all consecutive pairs of periods in parallel. We then propose a solution which overcomes this drawback in the following.

Shuffle-free Alignment Computation

The nearest-neighbor alignment computation approach we propose in this section takes advantage of the possibility to distribute the alignment workload more efficiently to favor local computations whenever possible. In order to avoid redundant computations, this method is composed of two steps.

The first step computes the similarities among the topics of each period and identifies for each topic its nearest neighbor in the same period. All the T_i topic sets are distributed such that each machine of the cluster receives one or more distinct T_i and computes the corresponding set of similarity measures S_{T_i} (Equation 3.2). For example, with 4 machines and 12 periods, assuming a round robin [95] distribution, M_1 could receive T_1 then T_5, then T_9.

The second step computes the topic alignments between consecutive periods. Each machine sends all its topic sets T_i as well as their related nearest neighbor similarities $S^{nn}_{T_i}$ (the subset of S_{Ti} restricted to the nearest neighbor of every $t \in T_i$) to the machine that contains T_{i+1}. Each node then locally computes the similarities between T_i and T_{i+1} and selects the alignments that satisfy the nearest-neighbor alignment conditions.

An important advantage of this parallel approach is that it is shuffle-free since all similarity computations are performed locally on each machine. The only data that is transferred to every machine are T_i, $S^{nn}_{T_i}$, and T_{i+1} with $S^{nn}_{T_i}$ being small since its size is linear to $|T_i|$.

This method is able to handle very large graphs as long as a single machine can handle the computation of a pair of periods. This meets a typical use case where the text corpus grows over time and gets more periods to analyse while the average size of a period remains fixed, *i.e.*, the number of topics in a period is controlled by the LDA step.

Another advantage of this approach is that it efficiently reduces the computation cost by avoiding redundant similarity computation. More precisely, every similarity between any pair of topics is computed only once. In particular, between two consecutive periods, it computes only $N_{sim2}(A)$ values. Overall, this saves $(P-2) \times N_{sim1}(A)$ similarity computations with respect to the method of Section 3.3.2.

Experiments

The following experiment compares the performance of both proposed nearest-neighbor alignment computation methods for large multi-period graphs.

We compute the alignments of topics along five periods for an increasing number of topics per period. We compare the response time of the two methods : the period-ordering alignment method on Page 45 which implies a lot of data shuffling and the shuffle-free alignment method. We run the experiments using 4 worker machines such that the shuffle-free method can compute one pair of periods per machine and utilizes the same number of CPU core as the period-ordering method.

In Figure 3.9, we report the response time of both methods for a varying number of topics. The relative benefit of the shuffle-free method is increasing with the number of topics and is up to 4 times faster for 10 000 topics. The main reason is because the shuffle-free method does not shuffle any data while the period-ordering method shuffles up to $30GB$ for 10 000 topics, as reported on Figure 3.10.

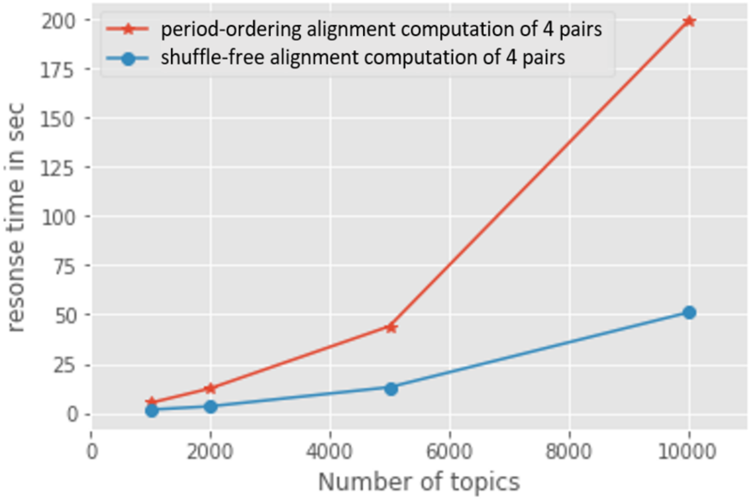


Figure 3.9: Similarity for 4 pairs of periods: response time vs. number of topics

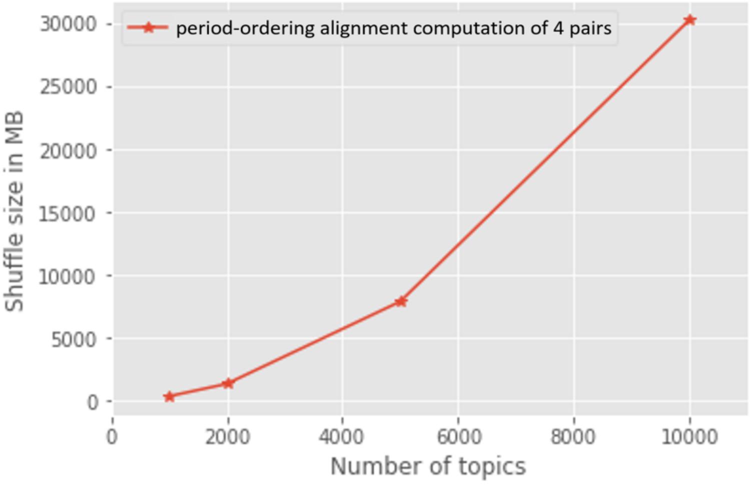


Figure 3.10: Similarity for 4 pairs of periods: shuffle size vs. number of topics

4 Pivot Graph Model and Query Language

This chapter presents the topic evolution model implemented in the EPIQUE workflow. The model is based on a multi-stage graph representation of topic evolution networks and introduces the notion of pivot evolution graphs to model the evolution of individual topics.

Contents

4.1 Topic Evolution Model

The motivation of our topic evolution model is based on the observation that the exploration of phylomemies is an interactive process where experts often want to explore the evolution starting from some relevant topics (*pivot topics* in our model) and analyze the subgraphs (*pivot graphs* in our model) around these topics. This exploration is guided by the detection of topic evolution patterns, which might be simple paths connecting topics, but also more complex evolution subgraphs representing the split of a topic into several sub-topics or the fusion of several topics into a single topic. Our solution to explore topic evolution graphs is not to produce a single phylomemy over the whole document corpus but to generate many subgraphs representing evolution of topics at different levels of detail.

Figure 4.1 explains the main ideas of our topic evolution model. Topic-term vectors extracted by *LDA* from different periods are aligned with an appropriate similarity measure to produce a complete alignment graph *Sim* (central part). Then the complete alignment

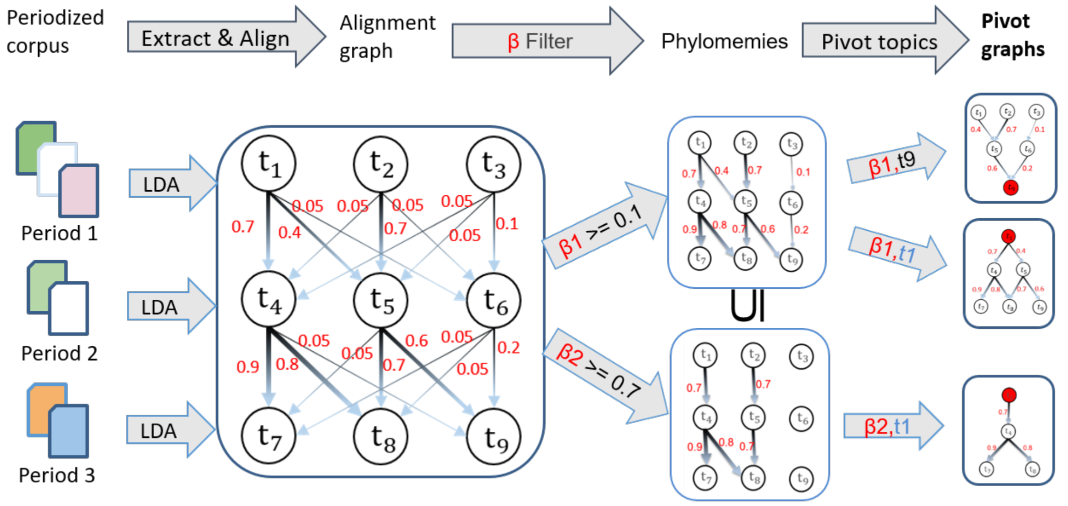


Figure 4.1: Topic evolution model of EPIQUE workflow

graph *Sim* can be studied by applying different similarity thresholds, called β thresholds ($\beta \in [0,1[$), to extract several phylomemetic graphs $\mathcal{G}$ connecting topics from subsequent periods.

Figure 4.2 and Figure 4.3 give two examples of the phylomemies over the Glyphosate corpus with similarity thresholds $\beta \geq 0.1$ and $\beta \geq 0.9$ respectively. It can be easily seen that the threshold influences the details that can be observed. More precisely, a low β value (*e.g.*, $\beta \geq 0.1$ in Figure 4.2), can result in a complex phylomemy connecting all topics which makes it not easy to analyze the evolution of a given topic. Whereas a high value (*e.g.*, $\beta \geq 0.9$ in Figure 4.3), may generate a simple phylomemy which contains many small linear subgraphs containing strongly connected topics and a lot of isolated topics with no information about their evolution. In between, there are many possible phylomemies which allow to study the evolution of scientific domains at different levels of detail and choosing a single β threshold does not allow to find the best phylomemetic graph. Thus, we focus the study of our topic evolution model on subgraphs.

As shown in Figure 4.1, a phylomemetic graph $\mathcal{G}$ is transformed into several groups of subgraphs defined by a topic t and a set of alignment thresholds β_i ($\beta_i \in [0,1[$). Each group contains for each topic t and threshold β_i the maximal connected subgraph with threshold β_i containing t. This subgraph is called *pivot graph* and will be defined in Section 4.2. We only consider subgraphs with at least one edge and ignore isolated topics (single node graphs).

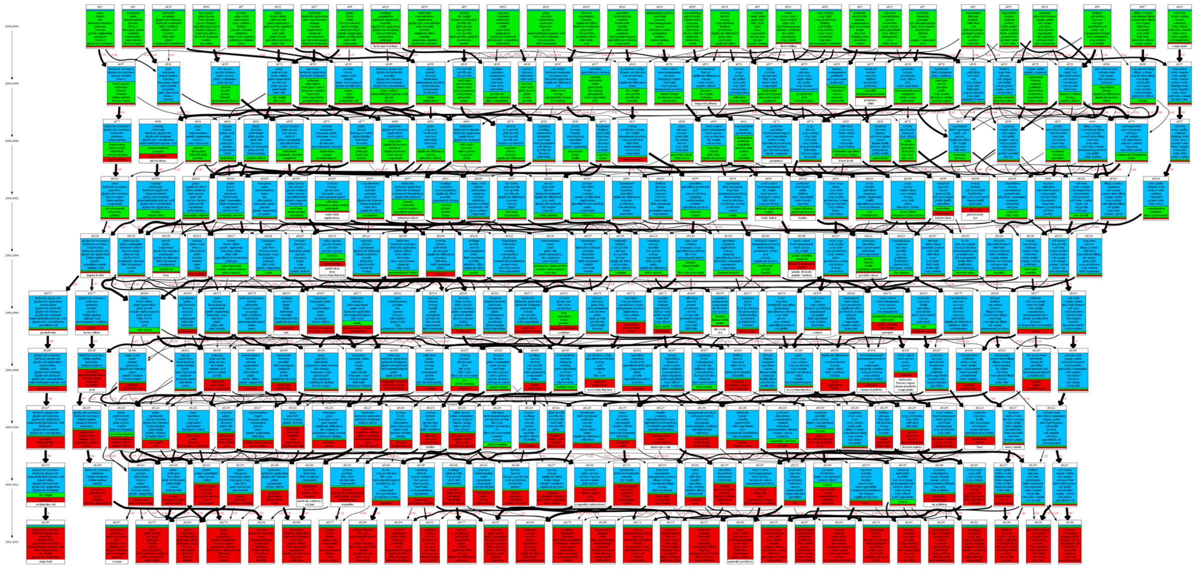


Figure 4.2: Phylomemy over the Glyphosate corpus with similarity threshold $\beta \geq 0.1$

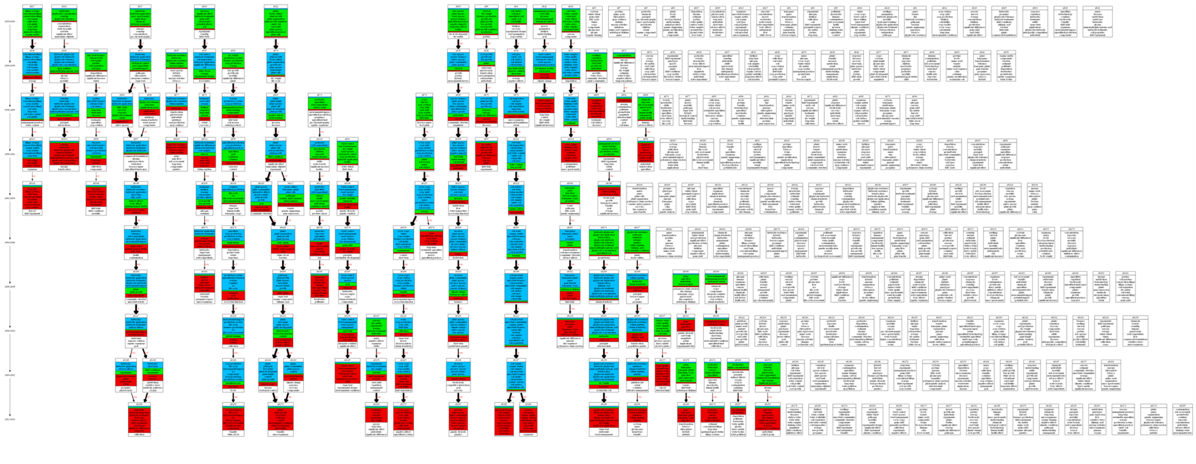


Figure 4.3: Phylomemy over the Glyphosate corpus with similarity threshold $\beta \geq 0.9$

The above process to transform a complete alignment graph *Sim* into groups of subgraphs mainly composes our topic evolution model as defined in Section 4.2. Then experts can use the *pivot topic query language* defined in Section 4.5 which consists of *pivot topic functions* (Section 4.3) and *pivot topic calculus* (Section 4.4) to query the resulting data of our workflow.

4.2 Pivot Topic Graphs

Definition 1 (topic) *Let V be a vocabulary of terms and P be an ordered sequence of time periods. A* topic *is a pair $t = (v, p)$ composed of a (sparse) weighted term vector $v \in \mathbb{R}^{|V|}$ and a period $p \in P$. We will denote by $t.terms$ the term vector and by $t.period$ the period of t.*

Definition 2 (topic evolution graph) *Let T be a set of topics, $sim : T \times T \rightarrow [0, 1]$ a similarity function estimating the semantic proximity of the term vectors of two topics. A* topic evolution graph *over T is a directed labeled* multistage *graph $\mathcal{G} = (T, E, sim)$ over T where the edges E connect all topics from consecutive periods with positive similarity values. That is, $E = \{(t_i, t_j) \in T | sim(t_i, t_j) > 0 \wedge t_j.period = t_i.period + 1\}$.*

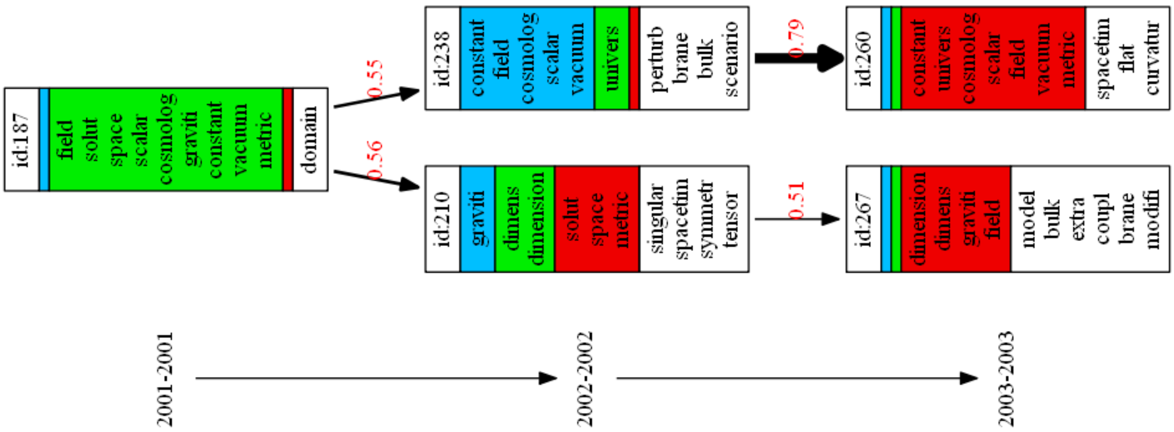


Figure 4.4: A subgraph of a topic evolution graph over the arXiv corpus

Example 3 *In Figure 4.4, P has 3 periods: p1="2001 − 2001", p2="2002 − 2002" p3="2003 − 2003", T_{p1} contains topic $t187 = (v, p1)$, where v is a weighted vector with positive weights for stemmed terms "field", "solut", "space", "scalar", etc. Whereas the term vector v of topic $t210 = (v, p2)$ contains "graviti", "dimens", "dimension", etc. Each topic is labeled by its top-10* *highest ranked terms and topic similarity is estimated by the cosine similarity between the corresponding two term vectors. For example, topic $t187$ and topic $t210$ have some stemmed terms in common, such as "solut", "space", "graviti" and "metric". And the cosine similarity between these 2 topics is $sim(t187, t210) = 0.56$.*

Topic evolution graphs only connect topics from two subsequent periods and it is not possible to directly connect two very similar topics t and t' from two distant periods, *i.e.*, $|t.period - t'.period| > 1$. This condition seems very restrictive and rejects a number of interesting evolution links. Whereas this restriction could safely be lifted without invalidating our approach, it also can be justified by several observations:

- Our model does not include any similarity threshold for connecting two topics. This means in particular that two similar but distant topics can still be connected by a path traversing some less similar topics (in Section 4.3 we introduce a graph evolution function $pevol^{\delta}$ which allows to compare any topic with any other reachable topic).
- Multistage graphs are visually more comprehensible than general directed acyclic graphs.
- The distance between topics also depends on the periodization scale and it is possible to generate evolution graphs with different granularity for the same document corpus.
- This restriction also reduces the size and complexity of evolution graphs and the corresponding computation costs and memory/disk space.

Analyzing topic evolution graphs is a complex task which includes various filtering operations for identifying topics by their terms, removing alignment edges below a certain threshold, selecting subgraphs with a specific structure etc. To solve this task, we propose a query language which allows users to extract *connected subgraphs* containing a given topic t and all alignment edges with some minimal similarity value β. This decomposition allows to formulate high-level filters for characterizing the semantic (labels) and structural (split, merge) evolution of topics in time.

Definition 3 (pivot topics and pivot evolution graphs) *Let t be a topic in some topic evolution graph $\mathcal{G} = (T, E, sim)$ and $\beta \in [0, 1]$. The pair (t, β) is called a* pivot topic *of t with similarity threshold β. Then, a* connected subgraph *$\mathcal{G}(t, \beta) = (T', E', sim, \beta)$ of $\mathcal{G}$ is a* pivot (evolution) graph of pivot topic (t, β) *if $t \in T'$ and $sim(t', t'') \geq \beta$ for all similarity edges $(t', t'') \in E'$, where $t' \in T'$ and $t'' \in T'$*

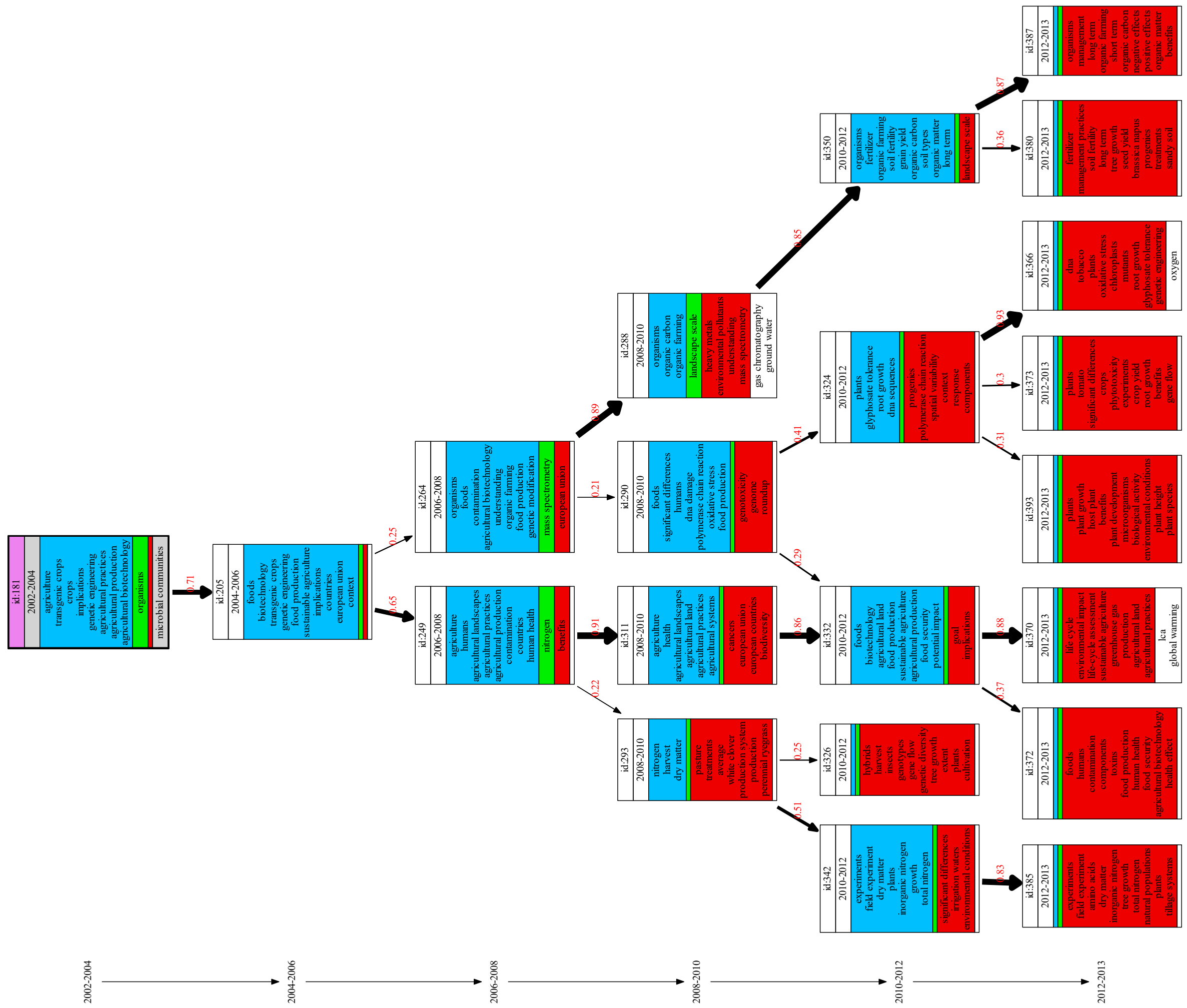


Figure 4.5: Pivot graph $\mathcal{G}(t181, 0.2)$ over the Glyphosate corpus

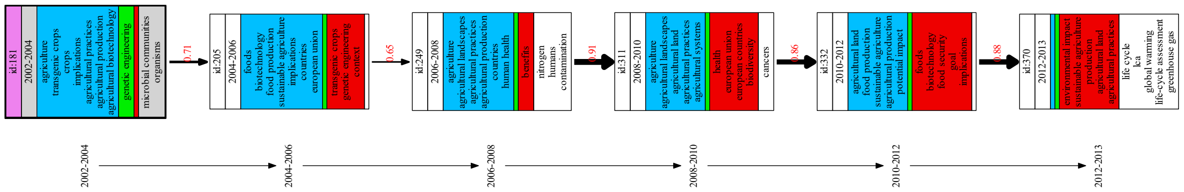


Figure 4.6: Pivot graph $\mathcal{G}(t181, 0.6)$ over the Glyphosate corpus

Example 4 *Figure 4.5 and Figure 4.6 present two different pivot graphs $\mathcal{G}(t181, 0.2)$ and $\mathcal{G}(t181, 0.6)$ extracted from the Glyphosate corpus for the same topic $t181$ of which the id field is highlighted in purple. Observe that the pivot graph $\mathcal{G}(t181, 0.6)$ contains a single path $t181 \rightarrow t205 \rightarrow t249 \rightarrow t311 \rightarrow t332 \rightarrow t370$, which is a subgraph of the pivot graph $\mathcal{G}(t181, 0.2)$.*

We distinguish three particular pivot evolution graphs among all possible pivot graphs of a given pivot topic (t, β) which allows users to study the evolution of the given pivot topic *wrt.* its future subgraph, past subgraph or both of them:

Definition 4 (future, past and history of a pivot topic) *The* future *of some pivot topic (t, β) is the* maximal *pivot evolution graph $\mathcal{G}^{future}(t, \beta)$ which contains all paths with* source *t. Equivalently, the* past *of some pivot topic (t, β) is the* maximal *pivot evolution graph $\mathcal{G}^{past}(t, \beta)$ which contains all paths with* target *t. The union of the past and future $\mathcal{G}^{*}(t, \beta) = \mathcal{G}^{past}(t, \beta) \cup \mathcal{G}^{future}(t, \beta)$ is called the* history *of pivot topic (t, β).*

Example 5 *Figure 4.7 shows the past of the pivot topic $(t181, 0.6)$. The future of the same pivot topic can be seen in Figure 4.6 of Example 4.*

By Definition 3, since $\beta_i \in [0, 1]$ is a real number, for each topic t there exists an infinite number of pivot topics (t, β_i). However, the number of pivot topics with different pivot histories is finite and depends on the distribution of the similarity values in the topic evolution graph. This observation is formalized in the following definition and proposition.

Definition 5 (topic spectrum) *Let $\mathcal{G}^{*}(t, 0) = (T, E, sim)$ be the* complete history of *t (the maximal connected subgraph of $\mathcal{G}$ containig t) and $S(t) = \{sim(t, t') | (t, t') \in E\}$ be the set of distinct similarity values (edge labels) in $\mathcal{G}^{*}(t, 0)$. We call $S(t)$ the spectrum of t.*

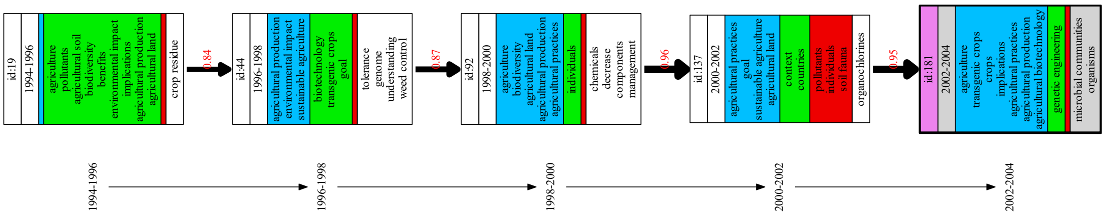


Figure 4.7: Pivot graph $\mathcal{G}^{past}(t181, 0.6)$ over the Glyphosate corpus

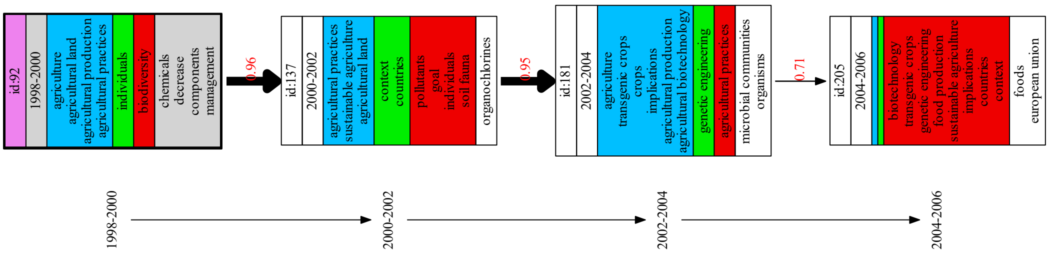


Figure 4.8: Pivot graph $\mathcal{G}^{future}(t92, 0.7)$ over the Glyphosate corpus

Example 6 *The topic spectrum of the pivot graph $\mathcal{G}^{future}(t92, 0.7)$ in Figure 4.8 contains values* $0.96, 0.95$ *and* 0.71.

Proposition 1 *The number of distinct pivot histories $\mathcal{G}^*(t, \beta)$, $\beta \in [0, 1]$, of a topic t is smaller or equal to the size $|S(t)|$ of the topic spectrum of t.*

Proof 1 *Suppose that $S(t) = \{\beta_1, \beta_2, ..., \beta_n\}$ and $\beta_i < \beta_{i+1}$ for all $1 \leq i < n$. Then $n = |S(t)|$ and it is sufficient to show that $\mathcal{G}^*(t, \beta) = \mathcal{G}^*(t, \beta')$ for all β and β' where (1) $\beta_i \leq \beta < \beta' < \beta_{i+1}$. We apply a proof by contradiction. By Definition 4, $\mathcal{G}^*(t, \beta)$ is the maximal connected subgraph of $\mathcal{G}$ containing topic t where all edges (t', t'') have a weight $sim(t', t'') \geq \beta$. Then it is easy to see that for all β, β' where $\beta \leq \beta'$, $\mathcal{G}^*(t, \beta') \subseteq \mathcal{G}^*(t, \beta)$. Suppose that both histories $\mathcal{G}^*(t, \beta')$ and*

$\mathcal{G}^*(t,\beta)$ are different, i.e., $\mathcal{G}^*(t,\beta') \subset \mathcal{G}^*(t,\beta)$. Then there exists an edge (t_1,t_2) in $\mathcal{G}^*(t,\beta)$ where (2) $\beta \leq sim(t_1,t_2) < \beta'$. By $\beta' < \beta_{i+1}$ and Definition 5, we also obtain $\mathcal{G}^*(t,\beta_{i+1}) \subset \mathcal{G}^*(t,\beta')$, i.e., there exists another edge (t_3,t_4) in $\mathcal{G}^*(t,\beta')$ where (3) $\beta' \leq sim(t_3,t_4) < \beta_{i+1}$. From (1), (2) and (3) we obtain $\beta_i \leq \beta \leq sim(t_1,t_2) < \beta' \leq sim(t_3,t_4) < \beta_{i+1}$, i.e., there exist two edges in $\mathcal{G}^*(t,0)$ with two different similarity values in $[\beta_i,\beta_{i+1}[$ which is in contradiction with Definition 5.

4.3 Pivot Topic Functions

The goal of our pivot graph model is to define a query language which allows users to filter topics according to useful criteria concerning the evolution of the vocabulary and the structure of their pivot evolution graphs.

The first two pivot topic functions return the period and the label of a pivot topic. Both functions are independent of the β threshold, the future and the past of the pivot topic. Function period returns the topic period of the pivot topic:

$$period(t) = t.period \tag{4.1}$$

All topics $t \in T$ are labeled by a subset of terms $labels(t) \subseteq V$. For example, in our system $labels(t)$ returns the k first terms in vocabulary V ranked by the weighted vector $t.terms$:

$$labels(t) = top_k(V)\text{: top-k terms in } V \text{ ranked by vector } t.terms \tag{4.2}$$

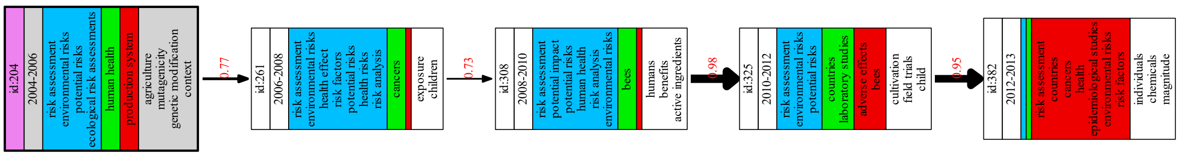


Figure 4.9: Pivot graph $\mathcal{G}^{future}(t204, 0.7)$ over the Glyphosate corpus

Example 7 *The pivot topic* $(t204, 0.7)$ *in Figure 4.9 contains the top-*10 *weighted terms, i.e., "risk assessment", "environmental risks, "potential risks", "ecological risk assessments", "human health", "production system", "agriculture", "mutagenicity", "genetic modification" and "context", as its label.*

The evolution of a topic t can be characterized by the structure of the future pivot evolution graph $\mathcal{G}^{future}(t, \beta)$ and the past pivot evolution graph $\mathcal{G}^{past}(t, \beta)$ of its pivot topics (t, β). In the following, let $\delta \in \{future, past\}$ and $\mathcal{G}^{\delta}(t, \beta)$ denote the past $(\delta = past)$ or the future $(\delta = future)$ of (t, β).

Path Functions The following function computes the set of topics that appear in the past and in the future of (t, β), respectively:

$$path^{\delta}(t, \beta) = \{t' | t' \text{ topic in } \mathcal{G}^{\delta}(t, \beta)\} \tag{4.3}$$

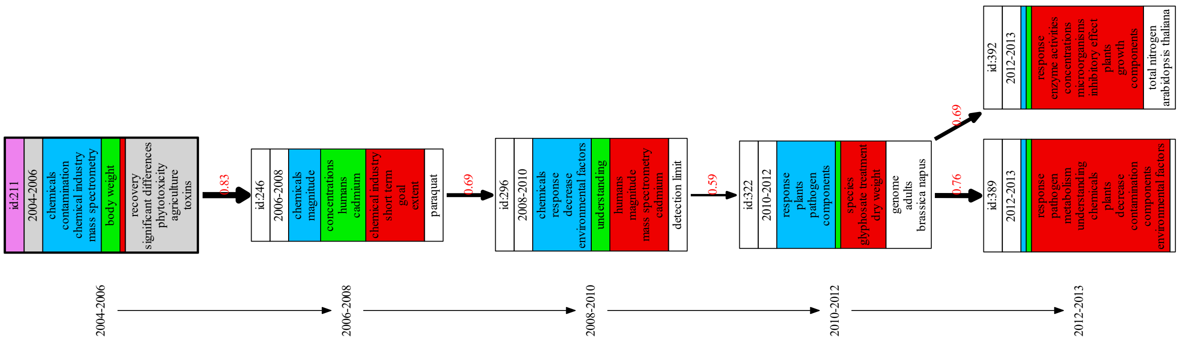


Figure 4.10: Pivot graph $\mathcal{G}^{future}(t211, 0.5)$ over the Glyphosate corpus

Example 8 *In Figure 4.10, topics* $t246, t296, t322, t389$ *and* $t392$ *appear in the future of pivot topic* $(t211, 0.5)$

Term Evolution Functions Term labels are useful to analyze the evolution of terms within the topic evolution graphs and to filter topics by their contents. For each pivot topic, we define the two sets $labels^{past}(t, \beta)$ and $labels^{future}(t, \beta)$ of topic labels which contain terms appearing, respectively, in the past and the future of pivot topic (t, β):

$$labels^{\delta}(t, \beta) = \{l | t' \in path^{\delta}(t, \beta) \wedge l \in labels(t')\} \tag{4.4}$$

Observe that $labels(t)$ is defined independently of β, whereas $labels^{future}(t, \beta)$ and $labels^{past}(t, \beta)$ might change for different β thresholds. The label terms of some pivot (t, β) can be classified into four disjoint categories:

- Emerging terms which exist in the future but not in the past:

$$emerge(t, \beta) = labels^{future}(t, \beta) - labels^{past}(t, \beta) \tag{4.5}$$

- Decaying terms which exist in the past but not in the future:

$$decay(t, \beta) = labels^{past}(t, \beta) - labels^{future}(t, \beta) \tag{4.6}$$

- Stable terms which exist in the past and the future:

$$stable(t, \beta) = labels^{future}(t, \beta) \cap labels^{past}(t, \beta) \tag{4.7}$$

- Specific terms which neither exist in the past topics nor in the future topics of pivot topic (t, β):

$$specific(t, \beta) = t.labels - (labels^{future}(t, \beta) \cup labels^{past}(t, \beta)) \tag{4.8}$$

Example 9 *The label terms of the pivot topic $(t204, 0.7)$ in Figure 4.9 are classified into four categories with "risk assessment", "environmental risks, "potential risks" and "ecological risk assessments" as stable terms, "human health" as emerging term, "production system" as decaying term, and "agriculture", "mutagenicity", "genetic modification" and "context" as specific terms.*

Graph Evolution Functions The *δ-liveliness* $live^{\delta}(t, \beta)$ is defined by the diameter of its pivot graph $\mathcal{G}^{\delta}(t, \beta)$.

$$live^{\delta}(t, \beta) = max\{length(path) | path \text{ is a path in } \mathcal{G}^{\delta}(t, \beta)\} \tag{4.9}$$

A high total liveliness value $live^{past}(t, \beta) + live^{future}(t, \beta)$ describes a long living topic. A past value $live^{past}(t, \beta) = 0$ means that topic t is emerging and a future value $live^{future}(t, \beta) = 0$ corresponds to a decaying topic.

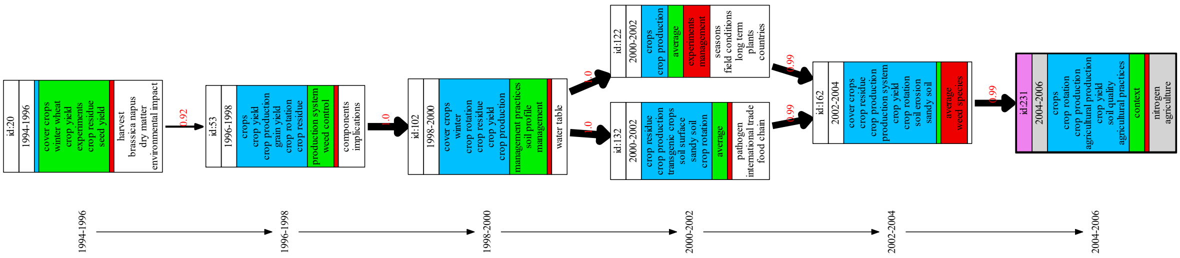


Figure 4.11: Pivot graph $\mathcal{G}^{past}(t231, 0.9)$ over the Glyphosate corpus

Example 10 *As can be seen in Figure 4.11, the liveliness value of the past of pivot topic* $(t231, 0.9)$ *is* $live^{past}(t231, 0.9) = 5$.

The *δ-relative evolution degree* $revol^{\delta}(t, \beta)$ is defined by the average topic dissimilarity (edge) weight in $\mathcal{G}^{\delta}(t, \beta) = (T, E, sim)$.

$$revol^{\delta}(t, \beta) = 1 - avg_{(t_i,t_j) \in E}(sim(t_i, t_j)) \tag{4.10}$$

A low relative evolution degree states that most topics evolve slowly in time whereas a high value signifies that most topics have an important “semantic gap”. By definition, we have $revol^{\delta}(t, \beta) \leq 1 - \beta$.

The *δ-pivot evolution degree* $pevol^{\delta}(t, \beta)$ is defined by the average dissimilarity of all topics in $\mathcal{G}^{\delta}(t, \beta) = (T, E, sim)$ with respect to the pivot topic t.

$$pevol^{\delta}(t, \beta) = 1 - avg_{t_i \in T}(sim(t, t_i))) \tag{4.11}$$

A low pivot evolution degree signifies that the pivot topic does not evolve much (all other topics are similar), whereas a high value indicates that the pivot topic evolves rapidly .

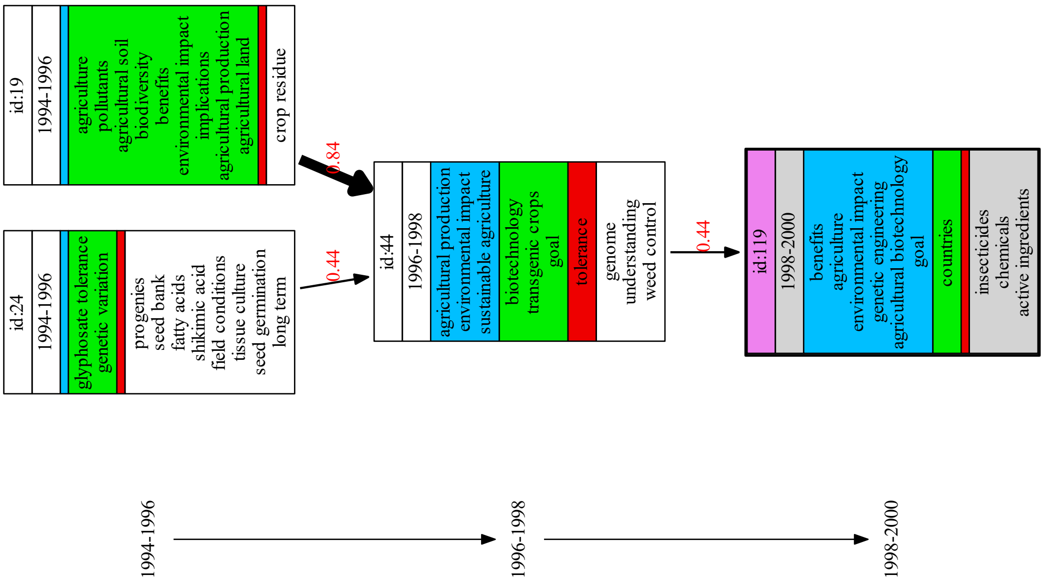


Figure 4.12: Pivot graph $\mathcal{G}^{past}(t119, 0.4)$ over the Glyphosate corpus

Example 11 *The relative evolution degree of pivot topic $(t119, 0.4)$ in Figure 4.12 is*

$$revol^{past}(t119, 0.4) = 1 - \frac{0.44 + 0.84 + 0.44}{3} = 0.427$$

whereas its pivot evolution degree is

$$\begin{aligned} pevol^{past}(t119, 0.4) =& 1 - \frac{sim(t24, t119) + sim(t19, t119) + sim(t44, t119)}{3} \\ =& \frac{sim(t24, t119) + sim(t19, t119) + 0.44}{3} \\ =& 0.707 \end{aligned}$$

The similarity values $sim(t24, t119)$ and $sim(t19, t119)$ are stored in the topic alignment graph Sim as mentioned in Section 4.1.

The *δ-split degree $split^{\delta}(t, \beta)$* is defined by the average outdegree of $\mathcal{G}^{\delta}(t, \beta) = (T, E, sim)$.

$$split^{\delta}(t, \beta) = \frac{|E|}{|\{t_i | t_i \in T \wedge (t_i, t_j) \in E\}|} \tag{4.12}$$

A low value signifies that the topics evolve along linear paths and a high value signifies that the topics split into several future sub-topics. Notice that the $split^{\delta}(t, \beta)$ function can be used for the future and the past of a given pivot topic (t, β), but it is more important for the future graph.

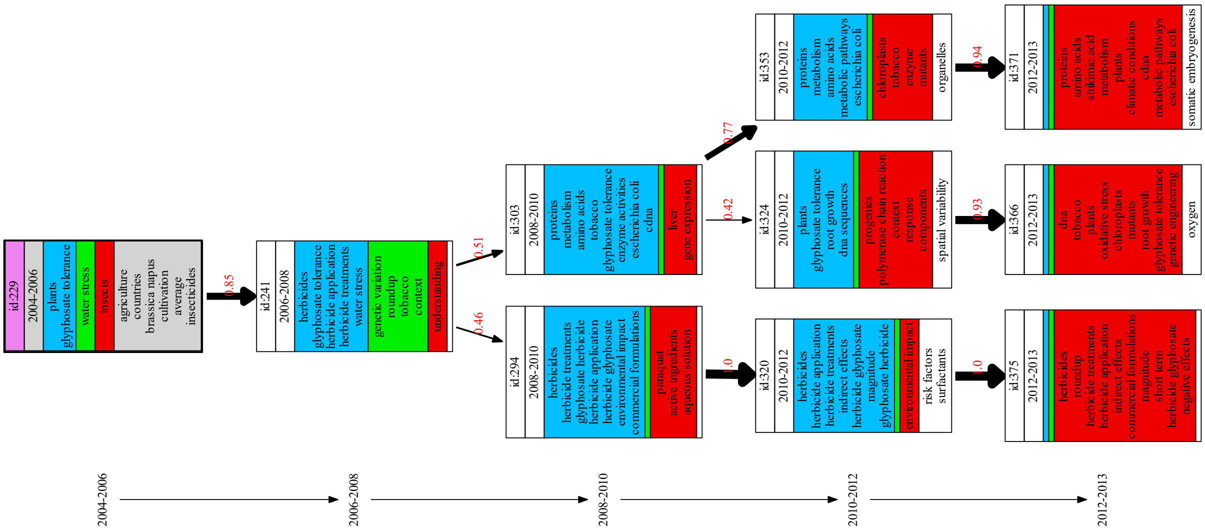


Figure 4.13: Pivot graph $\mathcal{G}^{future}(t229, 0.4)$ over the Glyphosate corpus

Example 12 *The split degree of pivot topic $(t229, 0.4)$ in Figure 4.13 is $split^{future}(t229, 0.4) = \frac{9}{7} = 1.286$.*

The *δ-convergence degree* $conv^{\delta}(t, \beta)$ is defined by the average indegree of $\mathcal{G}^{\delta}(t, \beta) = (T, E, sim)$.

$$conv^{\delta}(t, \beta) = \frac{|E|}{|\{t_j | t_j \in T \wedge (t_i, t_j) \in E\}|} \tag{4.13}$$

A low value signifies that many topics depend on a single parent topic and a high value signifies that many topics are the result of the fusion of past topics. Thus, this function is more important for the past graph of a pivot topic.

Example 13 *The convergence degree of pivot topic $(t92, 0.4)$ in Figure 4.14 is $conv^{past}(t92, 0.4) = \frac{3}{2} = 1.5$.*

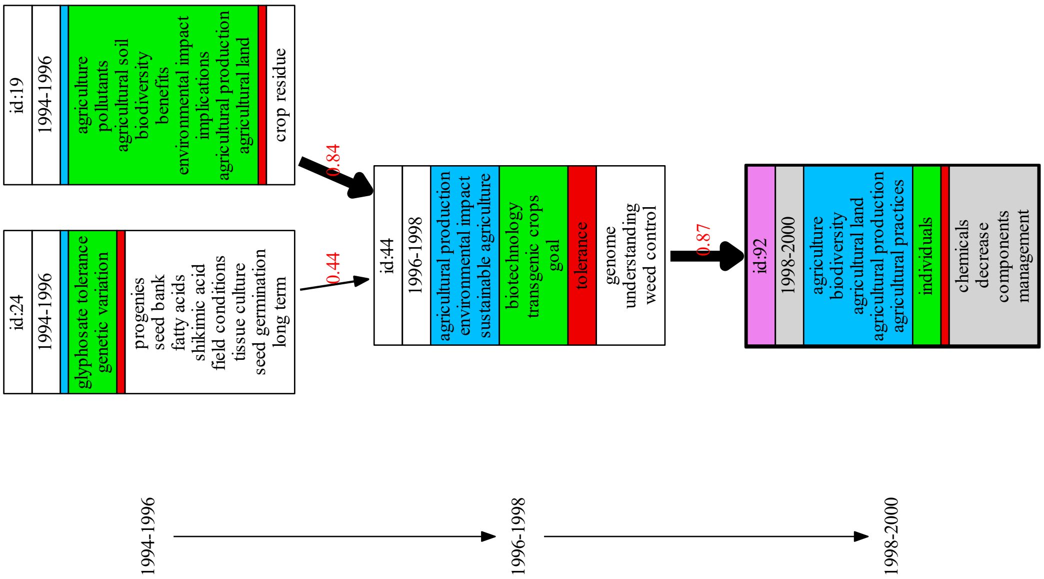
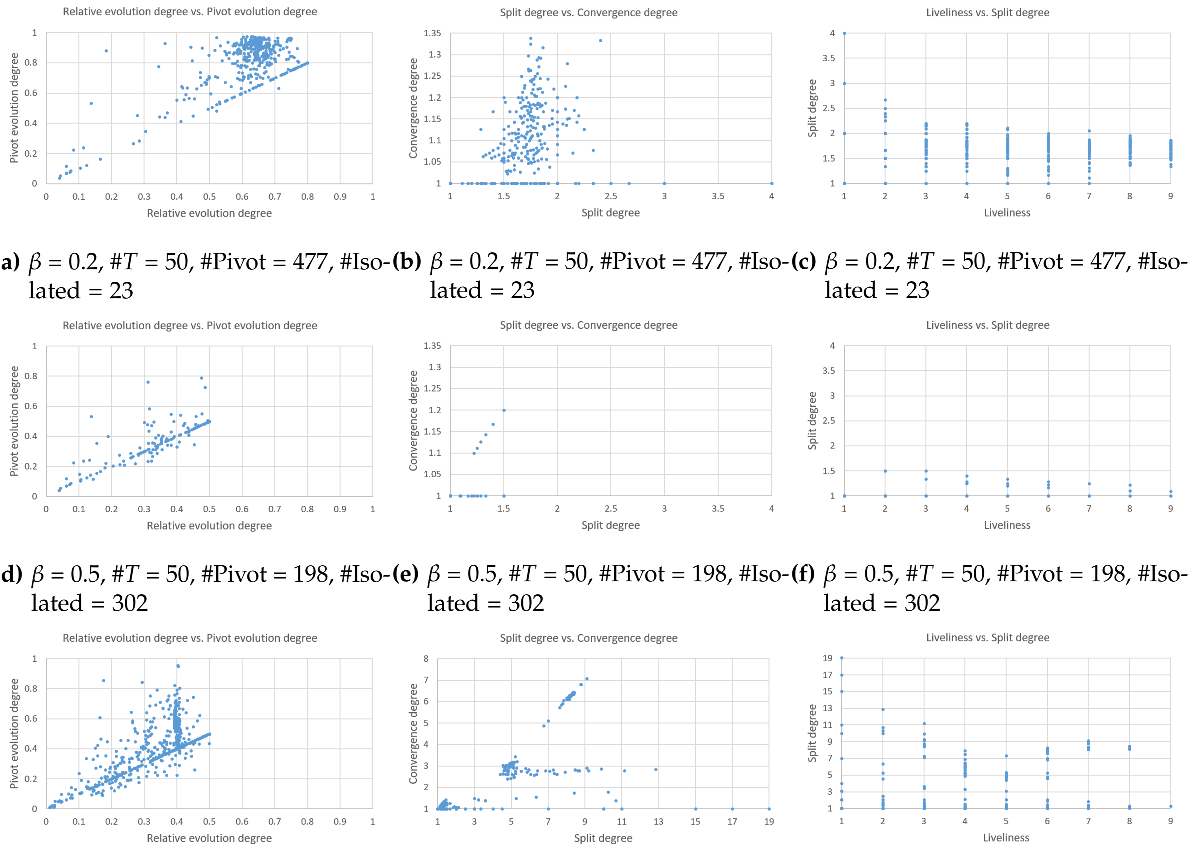


Figure 4.14: Pivot graph $\mathcal{G}^{past}(t92, 0.4)$ over the Glyphosate corpus

The previous functions can be used for the structural and quantitative analysis of the evolution of topics. The objective of the following paragraphs is to explore the impact of the main parameters, *i.e.*, the β threshold and the topic number $\#T$, on the structure and the semantics of the generated pivot evolution graphs.

Figure 4.15 shows the distribution of *future* pivot evolution graphs in arXiv *wrt*. three groups of metrics, the *relative evolution degree* vs. the *pivot evolution degree* , the *split degree* vs. *convergence degree* and the *liveliness* vs. the *split degree*. The figure is organized into 3 lines of 3 sub-graphs where each line corresponds to identical fixed parameters β and $\#T$ and each sub-graph corresponds to a group of metrics. On the first line, we set $\beta = 0.2$ and $\#T = 50$. On the second line, $\#T$ remains the same ($\#T = 50$) whereas β is increased to $\beta = 0.5$. On the third line, β remains the same as in the 2nd line ($\beta = 0.5$) whereas $\#T$ is increased to $\#T = 150$. Each Figure only shows pivot topic graphs with at least two nodes and the number of isolated topics is reported in the figure captions.

When comparing Figure 4.15a with Figure 4.15d, we can see that for the lower threshold $\beta = 0.2$, pivot topics evolve more than for the higher value $\beta = 0.5$. Lower β values also allow



(a) β = 0.2, #T = 50, #Pivot = 477, #Isolated = 23

(b) β = 0.2, #T = 50, #Pivot = 477, #Isolated = 23

(c) β = 0.2, #T = 50, #Pivot = 477, #Isolated = 23

(d) β = 0.5, #T = 50, #Pivot = 198, #Isolated = 302

(e) β = 0.5, #T = 50, #Pivot = 198, #Isolated = 302

(f) β = 0.5, #T = 50, #Pivot = 198, #Isolated = 302

(g) β = 0.5, #T = 150, #Pivot = 1014, #Isolated = 486

(h) β = 0.5, #T = 150, #Pivot = 1014, #Isolated = 486

(i) β = 0.5, #T = 150, #Pivot = 1014, #Isolated = 486

Figure 4.15: Distribution of future pivot evolution graphs in arXiv with respect to three groups of metrics by varying β and #T.

pivot topics to connect with more topics than higher β values which only connect similar topics. This is shown in Figure 4.15b which represents a large number of complex pivot topic graphs with higher split and convergence degrees than the pivot topic graphs in Figure 4.15e. The previous observation is also confirmed in Figure 4.15c and Figure 4.15f which compare topic *liveliness* vs. *split degree*: the lower threshold $\beta = 0.2$ generates pivot graphs which are more complex than pivot graphs with the same liveliness scores generated by $\beta = 0.5$. Therefore, for a fixed $\#T$, varying β allows for revealing interesting evolution patterns at different levels of detail where the evolution of some topic might be too complex for low β values and become more intelligible for higher β values.

When the topic number per period increases ($\#T = 150$ in Figures (g), (h) and (i)), the workflow generates more pivot graphs, some of which become very complex. For example, in Figure 4.15g, pivot topics tend to evolve a lot even for a low relative evolution degree. The pivot graphs in Figure 4.15h are much more complex than the graphs generated by the same β-threshold with $\#T = 50$ topics (Figure 4.15e and Figure 4.15f). As we can see, the *split degree* attains a value of 19 compared with maximal *split degree* 1.5 in Figure 4.15e. The increase of $\#T$ reduces the proportion of isolated topics, 30% for $\#T = 150$ compared with 60% for $\#T = 50$. This is due to the existence of many similar topics in each period, which also increases the probability that two topics can be aligned.

Monotonic Pivot Functions: A pivot topic function f can be monotonic with respect to β. More precisely, if for all topics t and thresholds $\beta \leq \beta'$ (if f returns a set, then $\leq$ corresponds to $\subseteq$ and $\geq$ corresponds to $\supseteq$):

- $f(t, \beta) \leq f(t, \beta')$, then f is *monotonically increasing*;
- $f(t, \beta) \geq f(t, \beta')$, then f is *monotonically decreasing*;
- f is *non-monotonic* otherwise.

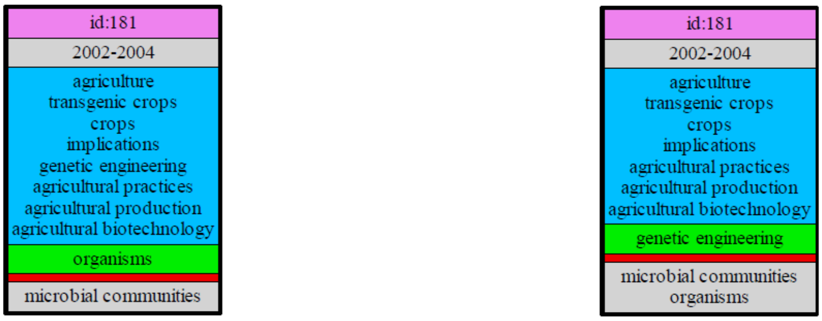
A function which is monotonically increasing *and* monotonically decreasing is called *constant* ($f(t, \beta) = f(t, \beta')$ for all β, β').

Table 4.1 outlines the monotonicity of each pivot topic function. The "monotonicity" of each function depends on the fact that $\mathcal{G}^{past}(t, \beta')$ is a subgraph of $\mathcal{G}^{past}(t, \beta)$ and $\mathcal{G}^{future}(t, \beta')$ is a subgraph of $\mathcal{G}^{future}(t, \beta)$ for all $\beta \leq \beta'$. Function $live^{\delta}$ is monotonically decreasing since for any t and $\beta \leq \beta'$, pivot graph $\mathcal{G}(t, \beta')$ is a subgraph of $\mathcal{G}(t, \beta)$ with a diameter

Table 4.1: Monotonicity of pivot topic functions

Function	Monotonicity	Function	Monotonicity
$period(t)$	constant	$path^{\delta}(t,\beta)$	monotonically decreasing
$labels(t)$	constant	$live^{\delta}(t,\beta)$	monotonically decreasing
$labels^{\delta}(t,\beta)$	monotonically decreasing	$revol^{\delta}(t,\beta)$	non-monotonic
$emerge(t,\beta)$	non-monotonic	$pevol^{\delta}(t,\beta)$	non-monotonic
$decay(t,\beta)$	non-monotonic	$split^{\delta}(t,\beta)$	non-monotonic
$stable(t,\beta)$	monotonically decreasing	$conv^{\delta}(t,\beta)$	non-monotonic
$specific(t,\beta)$	monotonically increasing		

$live^{\delta}(t,\beta') \leq live^{\delta}(t,\beta)$. Term labeling function $specific$ is monotonically increasing, *i.e.*, if label l is specific for (t,β) (it does not appear in the past or the future of (t,β)), it is also specific for all (t,β') where $\beta \leq \beta'$. Whereas function $stable$ is monotonically decreasing for the reason that if label l is stable for (t,β'), it is also stable for all (t,β) where $\beta \leq \beta'$. Functions $labels$ and $period$ are constant (return the same local label and period independently of β). $path^{\delta}$, $labels^{future}$ and $labels^{past}$ functions are monotonically decreasing. And $emerge(t,\beta)$, $decay(t,\beta)$, $revol^{\delta}(t,\beta)$, $pevol^{\delta}(t,\beta)$, $split^{\delta}(t,\beta)$ and $conv^{\delta}(t,\beta)$ are non-monotonic functions.



(a) Pivot topic $(t181, 0.2)$ of Figure 4.5

(b) Pivot topic $(t181, 0.6)$ of Figure 4.6

Figure 4.16: Label comparison of topic $t181$ *wrt.* different β values

Example 14 *Figure 4.16 zooms in Figure 4.5 and Figure 4.6, and compare the labels for the same topic* $t181$ *wrt. different* β *values. As we have mentioned before, function stable is monotonically decreasing, thus* $stable(t181, 0.6) \subseteq stable(t181, 0.2)$ *(blue box). Whereas function* $specific$ *is monotonically increasing, therefore* $specific(t181, 0.2) \subseteq specific(t181, 0.6)$ *(gray box).* $emerge(t181, 0.2) \neq emerge(t181, 0.6)$ *since it is a non-monotonic function. These two pivot topics do not have decaying labels but the monotonicity of function decay is the same as function emerge.*

4.4 Pivot Topic Calculus

The graph and term evolution functions allow to characterize the degree and the complexity of the evolution of pivot topics (t, β). Combined with other filters on the topic labels and the graph structure, it is possible to filter pivot topics satisfying rich topic evolution patterns.

Definition 6 (pivot topic filters) *Let $t, t' \in T$ be two topics, M be a set of pivot topic functions, $\phi \in \{=, \leq, \geq, <, >\}$ be a set of comparison predicates, c be a numerical constant, l be a term (label):*

- *The expression $t' \in path^{\delta}$ is a* path filter *which is true for (t, β) if t' is connected to t by a path in the future ($\delta = future$) or the past ($\delta = past$) of (t, β) and false otherwise;*
- *The expression $l \in labels^{\delta}$ is a* term label filter *which is true for (t, β) if l is a local (δ is empty), future ($\delta = future$) or past ($\delta = past$) label of (t, β) and false otherwise;*
- *If $evol^{\delta}$ ($revol^{\delta}$ or $pevol^{\delta}$) is a graph evolution function, then $evol^{\delta}\, \phi\, c$ is a* graph evolution filter *which is true for (t, β) if $evol^{\delta}(t, \beta)\, \phi\, c$ and false otherwise;*
- *The expression $period\, \phi\, c$ is a* period filter *which is true for a topic (t, β) if $t.period\, \phi\, c$ and false otherwise;*
- *If P and P' are pivot topic filters, then $P \wedge P'$, $P \vee P'$ and $\neg P$ are complex pivot topic filters with truth values following the semantics of the logical connectors $\wedge$, $\vee$ and $\neg$.*

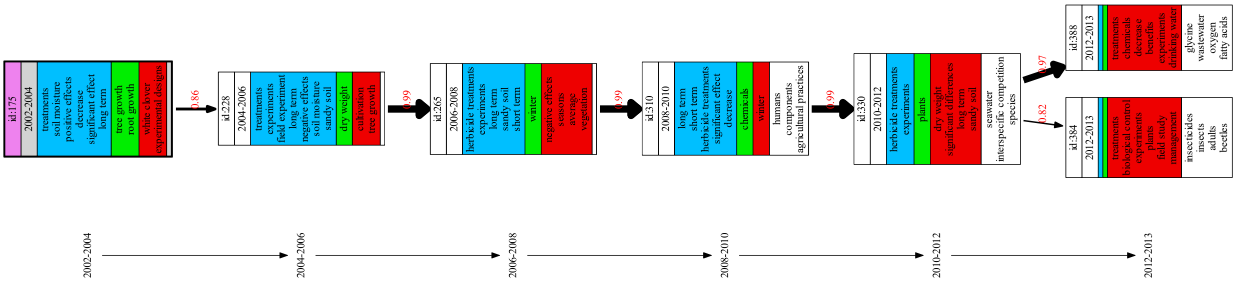


Figure 4.17: Pivot graph $\mathcal{G}^{future}(t175, 0.8)$ over the Glyphosate corpus

Example 15 *The pivot graph $\mathcal{G}^{future}(t175, 0.8)$ in Figure 4.17 is filtered by the following pivot topic* *filters:*

$$period \geq 2002 \wedge live^{future} \geq 5 \wedge split^{future} \leq 1.2 \wedge revol^{future} \leq 0.1 \wedge pevol^{future} \geq 0.1$$

In the following we will name filters by the name of the topic functions they use. For example $L \subseteq labels$ is called a $labels$ filter.

Observe that we can then redefine the term label filters *emerge*, *decay*, $stable$ and $specific$ using $labels$, $labels^{past}$ and $labels^{future}$:

- *emerging* local terms which do not exist in past topics: $l \in emerge \equiv l \in labels \wedge labels^{future} \wedge l \notin labels^{past}$
- *decaying* local terms which do not exist in future topics: $l \in decay \equiv l \in labels \wedge labels^{past} \wedge l \notin labels^{future}$
- *stable* local terms which exist in the past and the future topics of t: $l \in stable \equiv l \in labels \wedge labels^{future} \wedge labels^{past}$
- *specific* local terms which neither exist in the past nor in the future topics of t: $l \in specific \equiv l \in labels \wedge l \notin labels^{future} \wedge l \notin labels^{past})$

Pivot Filter Monotonicity: The monotonicity property of pivot functions presented in Section 4.3 can directly be transposed to pivot topic filters. More precisely, for all monotonically increasing functions F, predicate $pred = l \in F$ (F returns a set) or predicate $pred = F \geq c$ (F returns a number) are monotonic[1]. Symmetrically, for all monotonically decreasing functions F, predicate $pred = l \in F$ (F returns a set) or predicate $pred = F \geq c$ (F returns a number) are anti-monotonic[2]. Finally, if F is constant or non-monotonic, the corresponding filters are static or non-monotonic. Table 4.2 summarizes this categorization for each function.

Example 16 *The filter predicate $specific$ is monotonic since it is the conjunction of two monotonic predicates: $specific \equiv labels \wedge \neg(labels^{future} \vee labels^{past})$ (observe that $l \in labels^{future} \vee l \in labels^{past}$ is anti-monotonic and its negation $l \notin labels^{future} \wedge l \notin labels^{past}$ is monotonic). The filter predicate stable is anti-monotonic since it is the conjunction of a non monotonic and two anti-monotonic predicates. The two term evolution predicates emerge and decay are conjunctions of a*

[1] For all (t, β), if $l \in F$ holds for (t, β) then $l \in F$ holds for all (t, β') where $\beta' \geq \beta$.

[2] For all (t, β), if $l \in F$ holds for (t, β) then $l \in F$ holds for all (t, β') where $\beta' \leq \beta$.

Table 4.2: Monotonicity of complex filter predicates

P	P'	$P \wedge P'$	$P \vee P'$	$\neg P$
monotonic	monotonic	monotonic	monotonic	anti-monotonic
monotonic	anti-monotonic	non monotonic	static	anti-monotonic
monotonic	non monotonic	non monotonic	monotonic	anti-monotonic
monotonic	static	monotonic	static	anti-monotonic
anti-monotonic	anti-monotonic	anti-monotonic	anti-monotonic	monotonic
anti-monotonic	non monotonic	non monotonic	anti-monotonic	monotonic
anti-monotonic	static	anti-monotonic	anti-monotonic	monotonic
non monotonic	non monotonic	non monotonic	non monotonic	non monotonic
non monotonic	static	non monotonic	static	non monotonic
static	static	static	static	static

monotonic and an anti-monotonic predicate and, therefore non monotonic (if $l \in emerge$ is true for (t, β) it can be true or false for (t, β') with $\beta' > \beta$ or $\beta' < \beta$).

4.5 Pivot Topic Query Language

Let $DB(T, sim) = \{(t, \beta) | t \in T \wedge \beta \in S(t)\}$ be the set of pivot topics defined by a set of topics T and a similarity function sim. We call DB the *pivot database* defined by T and sim. In the following, we define a query language for extracting pivot topic subsets $Pivots \subseteq DB$. The semantics of this query language is based on the pivot topic calculus we have introduced in Section 4.4.

Let L denote a set of terms and c be a numerical constant. Atomic filter expressions and their semantics are defined using the topic calculus as shown in Table 4.3a. The first column corresponds to the query expression, the second column defines the corresponding quantified filter predicate and the third column shows the monotonicity property of each filter for $\phi = \leq$. Filter expressions can be composed to build complex filter expressions as shown in Table 4.3b. Finally, the query language defines two projection operators **Future** and **Past** (Table 4.3c) which modify the future/past parameter δ ($\delta = future$ by default).

Definition 7 (pivot query) *Let DB be a pivot database (set of pivot topics), Q be a pivot filter and $I(\overline{Q}, t, \beta)$ be a truth-functional interpretation of the pivot topic predicate $\overline{Q}$ given pivot topic (t, β).*

The expression DB.Q is a query which returns all pivot topics in DB where $\overline{Q}$ is true.

$$DB.Q = \{(t,\beta)|(t,\beta) \in DB \wedge I(\overline{Q},t,\beta)\}$$

Expression F	Semantics $\overline{F}$	Monotonicity ($\phi = \leq$)
Contains(L)	$\exists l \in L : l \in labels$	static
Emerge(L)	$\exists l \in L : l \in emerge$	non monotonic
Decay(L)	$\exists l \in L : l \in decay$	non monotonic
Stable(L)	$\exists l \in L : l \in stable$	anti-monotonic
Specific(L)	$\exists l \in L : l \in specific$	monotonic
Live ϕ c	$live^{\delta}\ \phi\, c$	anti-monotonic
Revol ϕ c	$revol^{\delta}\ \phi\, c$	non monotonic
Pevol ϕ c	$pevol^{\delta}\ \phi\, c$	non monotonic
Split ϕ c	$split^{\delta}\ \phi\, c$	non monotonic
Conv ϕ c	$conv^{\delta}\ \phi\, c$	non monotonic
Period ϕ c	$period\, \phi\, c$	static

(a) Atomic filter expressions

Expression F	Semantics $\overline{F}$	Monotonicity
$F_1.F_2$	$\overline{F_1} \wedge \overline{F_2}$	see Table 4.2
Minus(F_1)	$\neg\overline{F_1}$	see Table 4.2
F_1.**Union**(F_2)	$\overline{F_1} \vee \overline{F_2}$	see Table 4.2
Path(F_1)	$\exists t : t \models \overline{F_1} \wedge t \in path^{\delta}$	anti-monotonic

(b) Complex filter expressions

Expression	Semantics
Future	$\delta := f$
Past	$\delta := p$

(c) Graph projection

Table 4.3: Pivot Filter Expressions

Example 17 *Our query language allows to filter pivot topics according to some evolution pattern defined by the combination of graph evolution filters.*

- *For example query Q1 filters all pivot topics with high future relative and high pivot evolution degrees, where each future topic has two child topics in average and there exist future subtopics related to the pivot topic with a minimal distance of 5 periods:*

  *Q1: DB.**Future**.**Revol**(>=0.5).**Pevol**(>=0.6).**Split**(>=2).**Live**(=5)*

$$\overline{\text{Q1}} : revol^{future} \geq 0.5 \wedge pevol^{future} \geq 0.6 \wedge split^{future} \geq 2 \wedge live^{future} = 5$$

Parameter δ is by default equal to $future$.

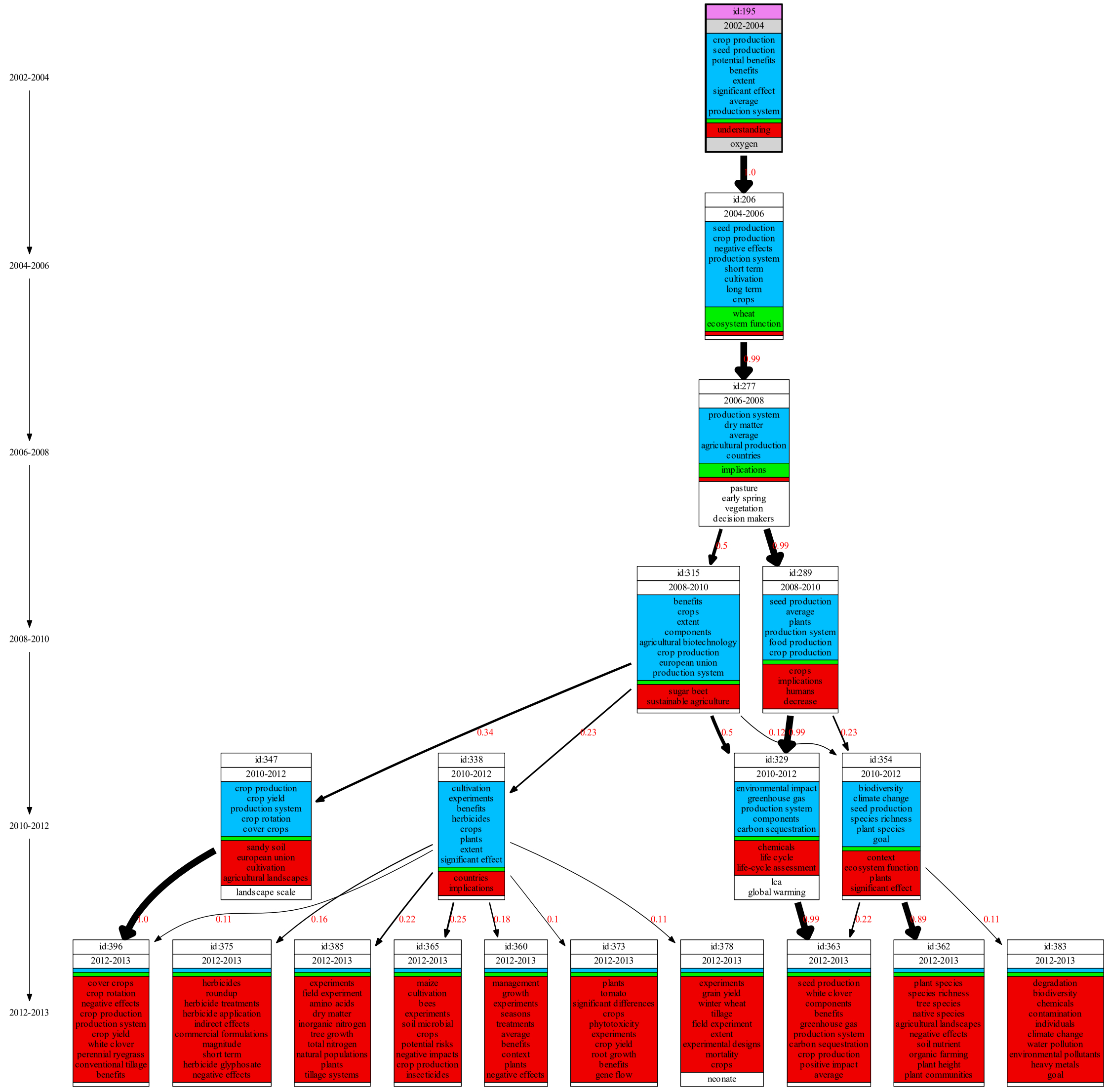


Figure 4.18: $\overline{Q1} : revol^{future} \geq 0.5 \wedge pevol^{future} \geq 0.6 \wedge split^{future} \geq 2 \wedge live^{future} = 5$

The result of query Q1 is shown in Figure 4.18. There are three other examples as displayed in Figures 4.19 to 4.21. Observe that the user does not specify the β-threshold. Although Figure 4.19 and Figure 4.21 have the similar structure, they have different evolution pace (corresponding to different β values). The pivot graph in Figure 4.19 has more emerging terms (green part) whereas the pivot graph in Figure 4.21 has more stable terms (blue part) which correspond to our queries to select high-evolution and low-evolution pivot topics respectively. Compared with the pivot graph of topic 178 (Figure 4.21), pivot topic 257 (Figure 4.20) has a shorter liveliness value and a little bit more complex evolution structure.

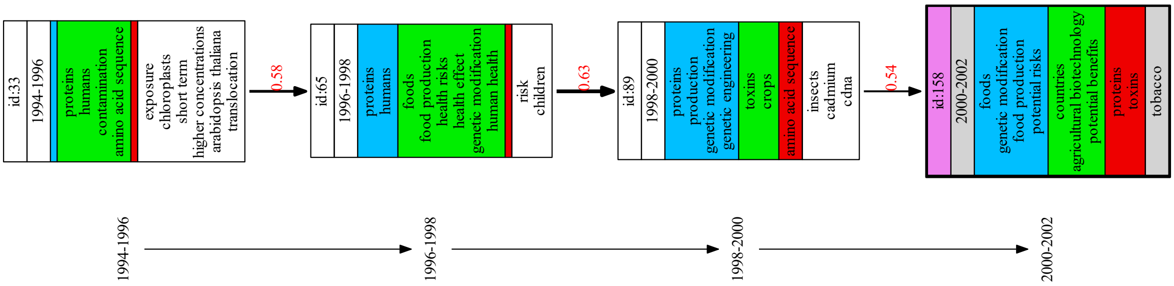


Figure 4.19: $\overline{Q2} : revol^{future} \geq 0.5 \wedge pevol^{future} \geq 0.6 \wedge split^{future} \leq 1.2 \wedge live^{future} = 3$

Apart from these graph structure-based filters, our query language also allows users to define other multi-dimensional filtering criteria including topic labels and temporal conditions for the selection of pivot topics:

- *Find all topics with stable term "algorithm" and without emerging term "learning":*

  *Q5: DB.**Stable**(*`"algorithm"`*).**Minus**(**Emerge**(*`"learning"`*))*

$$\overline{Q5} : \text{“algorithm”} \in stable \wedge \text{“learning”} \notin emerge$$

- *Find all topics with an emerging term "network" where the past contains a path to a topic with the stable term "protocol":*

  *Q6: DB.**Emerge**(*`"network"`*).**Past**.**Path**(**Stable**(*`"protocol"`*))*

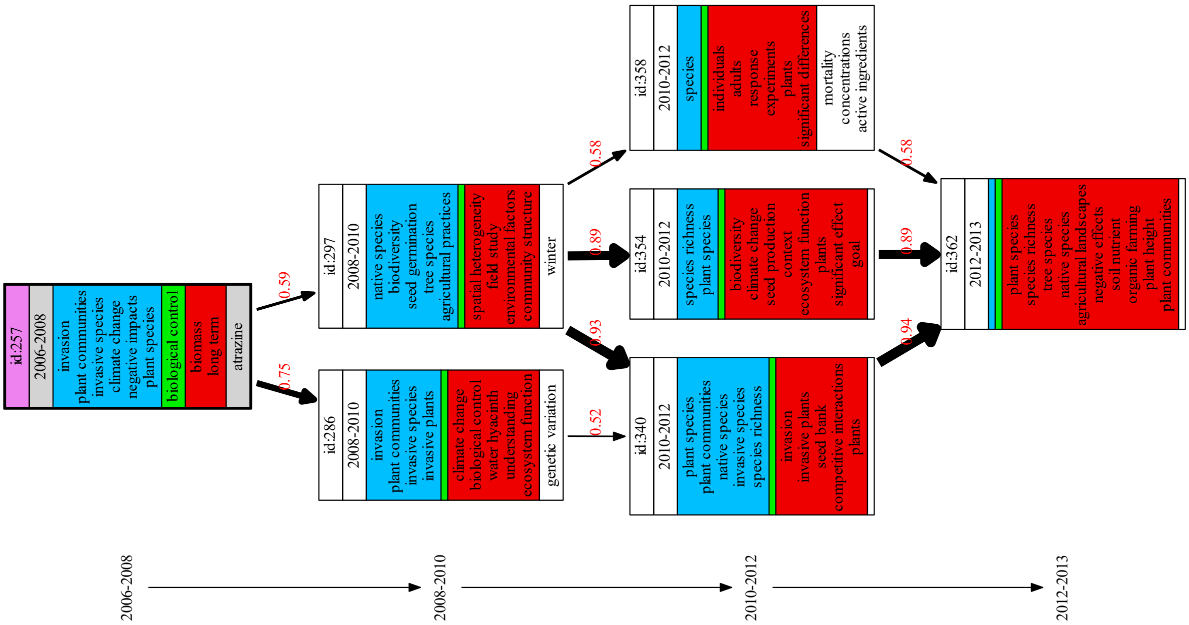


Figure 4.20: $\overline{Q3}$: $revol^{future} \leq 0.4 \wedge pevol^{future} \leq 0.5 \wedge split^{future} \geq 1.5 \wedge live^{future} = 3$

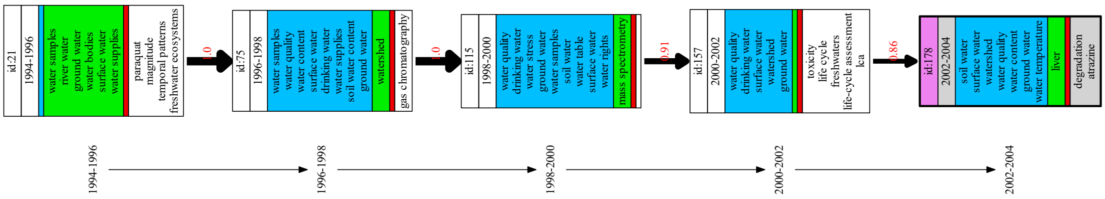


Figure 4.21: $\overline{Q4}$: $revol^{future} \leq 0.4 \wedge pevol^{future} \leq 0.5 \wedge split^{future} \leq 1.2 \wedge live^{future} = 4$

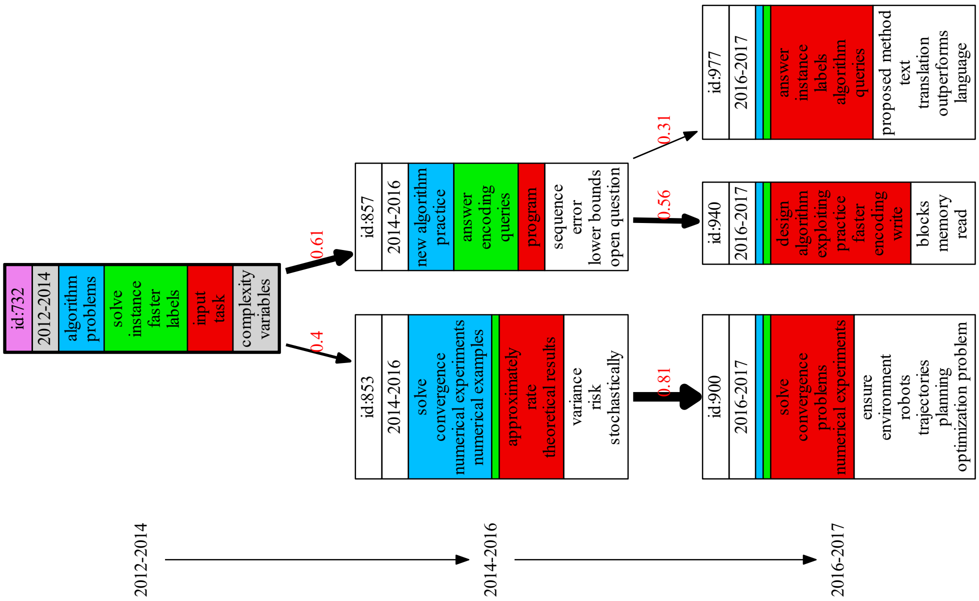


Figure 4.22: $\overline{Q5}$: “algorithm” $\in$ *stable* $\wedge$ “learning” $\notin$ *emerge*

$$\overline{Q6} : \textit{"network"} \in emerge.path^{past}(\textit{"protocol"} \in stable)$$

Observe that expression ***Past*** *changes parameter δ to past for the following sub-expression* ***Path****.*

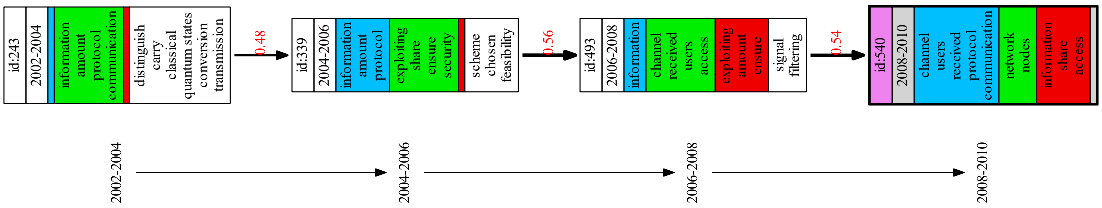


Figure 4.23: $\overline{Q6}$: "network" $\in emerge.path^{past}$("protocol" $\in stable$)

The results of queries Q5 and Q6 are illustrated in Figures 4.22 and 4.23.

Our query language allows users to select topics in specific regions of the sub-figures in Figure 4.15. For example query $Q1$ in Example 17 chooses all topics which appear in the upper right window of Figures 4.15a and on the right part of Figure 4.15b on the line corresponding to the liveliness value 5 in Figure 4.15c.

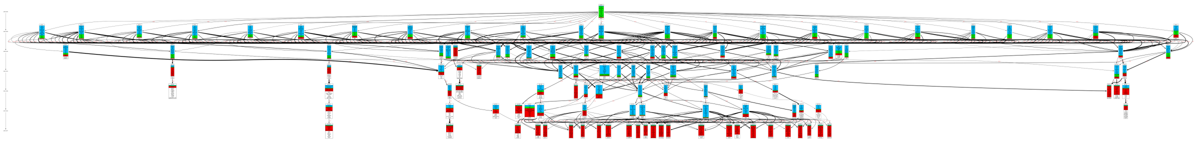


Figure 4.24: Visualization of pivot graph $\mathcal{G}^{future}(495, 0.5)$ where $\#T$ = 150

Figure 4.24 shows a future arXiv pivot graph generated for $\#T = 150$ and $\beta = 0.5$, which corresponds to a data point in Figure 4.15h where $split^{future}(t, 0.5)$ = 8.3 The graph connects topics with similarity higher than $\beta = 0.5$ and has nevertheless a high split and convergence degree. Figure 4.25 zooms into the rectangle of Figure 4.24 and we can observe that the topics in the second period are very similar which explains why the single root pivot topic is connected to more than 20 topics in the second period (this problem is solved by the diversity test described in Section 3.2.2 which reduces the number of topics per period).

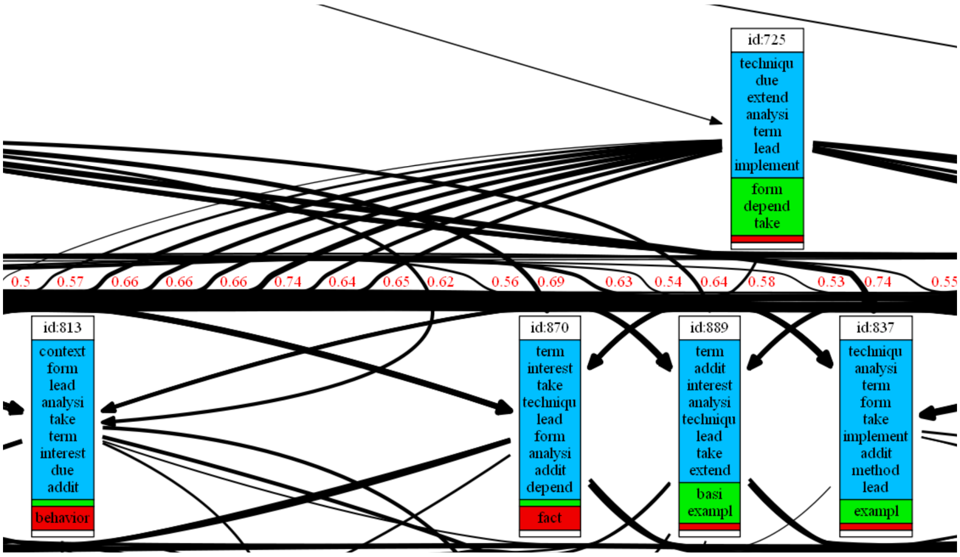


Figure 4.25: Zoom in Figure 4.24

5 Pivot Graph Generation

Our query evaluation strategy consists of precomputing all pivot graphs $\mathcal{G}^{future}(t, \beta_i)$ and $\mathcal{G}^{past}(t, \beta_i)$ for a sequence of thresholds β_0, β_1, ..., β_n where $\beta_i < \beta_{i+1}$ with the corresponding labels and graph evolution metrics. The computation of the pivot graphs is done once independently of any pivot topic queries and the generated tables can be processed using standard, non-recursive SQL queries for computing topic labels (Section 6.1.2) and graph evolution metrics (Section 6.1.3) and for evaluating pivot query expressions (Section 6.1.4).

This chapter will discuss the different pivot graph computation models that we proposed. Section 5.1 introduces three relational computation models where the main difference is the *Transitive Closure* computation algorithm that allows to compute paths for pivot graphs. The baseline algorithm (Section 5.1.1) uses the classic semi-naive approach to avoid recomputing TC paths that have been found at each β-iteration. The smart algorithm (Section 5.1.2) reduces the number of TC joins in a logarithmic manner at each β-iteration. And the incremental approach (Section 5.1.3) computes TC tables incrementally by reusing the results of the previous β-iteration. It is obvious that the relational pivot graph computation models provoke many join operations. We then propose a BSP-based [96] pivot graph computation model (Section 5.2) which applies the Pregel paradigm to generate pivot graphs. Experiments in Section 5.3 detail the performance evaluation of these models.

Contents

5.1 Relational Pivot Graph Computation

Our first implementation uses the relational data model to represent pivot graphs where a tuple (t, x, y, rs, ps, d, i) represents an edge (x, y, rs) in pivot graph $\mathcal{G}^{\delta}(t, \beta_i)$ of pivot topic (t, β_i) with relative evolution similarity rs, pivot evolution similarity ps of x $(Past)$ and y $(Future)$ and distance d of x $(Past)$ and y $(Future)$ from t.

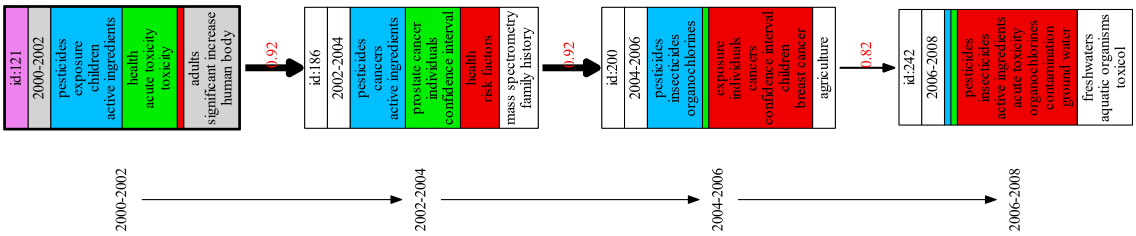


Figure 5.1: Pivot graph $\mathcal{G}^{future}(t121, 0.8)$ over the Glyphosate corpus

Example 18 *We use Figure 5.1 as an example. The table of pivot topic* $\mathcal{G}^{future}(t121, 0.8)$ *contains a tuple* $(t121, t186, t200, 0.92, 0.99, 2, 0.8)$ *representing that the edge* $(t186, t200, 0.92)$ *is reached by the pivot topic* $(t121, 0.8)$ *with the pivot evolution similarity* 0.99 *and the distance* 2 *of topic* $t200$ *from topic* $t121$.

We use a Datalog-like syntax for presenting three different algorithms to compute pivot graphs connecting all topics to all other topics reachable by a path. The main difference among these algorithms lies in the computation of the *transitive closure* (TC) edges which is the most important and complex pivot graph computation step.

The first (baseline) algorithm presented in Section 5.1.1 uses a semi-naive method to compute all TC edges for each β threshold separately. The second (smart) algorithm introduced in Section 5.1.2 improves the first algorithm by using a recursive-doubling technique [97] to reduce the number of TC joins at each β-iteration. Since the same TC edges may appear for different β values, the idea of avoiding the redundant computation brings about a more efficient incremental algorithm which is presented in Section 5.1.3. This algorithm avoids redundant computation by exploiting the monotonicity of pivot graphs where $\delta \in \{future, past\}$, $\mathcal{G}^{\delta}(t, \beta_{i+1})$ is a subgraph of $\mathcal{G}^{\delta}(t, \beta_i)$ for all $0 \leq i < n$. Both the baseline algorithm and the incremental algorithm are iteratively joining a partially computed table *TC* with the table of graph edges *Graph* until reaching the fixpoint. The implementation on top of Apache Spark of the incremental algorithm is discussed in Chapter 6.

5.1.1 Baseline Join-based TC computation

The baseline approach computes all pivot graphs for each beta threshold independently from the other thresholds. The transitive closure of alignment edges is obtained by a linear recursive rule. To avoid the recomputation of the same paths, we apply the semi-naive approach [98] which extends at each iteration only the paths that have been discovered in the previous iteration.

The input of the Datalog program (Algorithm 1) is defined by three tables:

- a topic table $Topics(t, p)$ storing the topic identifiers t with their periods p;
- a topic similarity matrix $Sim(t1, t2, s)$ where $t1$ and $t2$ are two topics and $s = sim(t1, t2)$ is the similarity between topic $t1$ and topic $t2$, and
- a sequence of β_i values defined as a binary table $Beta(b, i)$.

The first rule (Line 1) defines $Graph(x, y, s)$ which connects all topics x of period p to all topics y of period $p + 1$ where there exists an evolution edge of similarity $s = sim(x, y)$. Starting from *Graph*, the following recursive rules (Line 2 and Line 3) compute the transitive closure of the alignment subgraph of *Graph* for all topics in T and beta values β_i in $Beta(b, i)$

where all edges have a weight greater or equal to β_i and d is the graph distance between x and y[1]. Pivot topic evolution graphs are computed by first generating table $TC(x, y, d, i)$. Then tables *Future* and *Past* contain the pivot topic evolution graphs $\mathcal{G}^{\delta}(t, \beta_i)$ of all topics t, which correspond to the tuple structure as defined at the beginning of Section 5.1. Both tables are obtained by four non-recursive rules (Line 4 to Line 7).

Algorithm 1 Baseline pivot graph algorithm

```
1: Graph(x,y,s) ← Topics(x,p), Topics(y,p+1), Sim(x,y,s).
2: TC(x,y,1,i) ← Graph(x,y,s), Beta(b,i), s>=b.
3: TC(x,z,d+1,i) ← TC(x,y,d,i), Graph(y,z,s), Beta(b,i), s>=b.
4: Future(t,t,y,s,s,1,i) ← Graph(t,y,s), Beta(b,i), s>=b.
5: Future(t,x,y,rs,ps,d+1,i) ← TC(t,x,d,i), Graph(x,y,rs), Sim(t,y,ps).
6: Past(t,x,t,s,s,1,i) ← Graph(x,t,s), Beta(b,i), s>=b.
7: Past(t,x,y,rs,ps,d+1,i) ← TC(y,t,d,i), Graph(x,y,rs), Sim(x,t,ps).
```

5.1.2 Smart Join-based TC Computation

The linear transitive closure of the baseline approach can be computed much faster by a *recursive-doubling* technique [97]. Instead of joining the partially computed transitive closure TC table with the edges of the base graph *Graph*, each iteration of the recursive-doubling algorithm joins the transitive closure relation with itself. Thus, on a graph of diameter d, it needs only $log_2 d$ iterations, rather than d iterations before convergence. But the direct application of recursive-doubling method does a lot of redundant computation because it discovers the same TC edges several times. *Smart Transitive Closure* (Smart TC) independently proposed by [99] and by [100] is a variant of the recursive doubling method which avoids this redundant computation.

Intuitively, Smart TC avoids the repeated computation of the same edges by breaking each path into a *prefix*, whose length is a power of two and a *suffix* whose length is not greater than the length of the prefix. We use a table $Q(x, y, d, i, k)$ to represent the prefix table which contains all those pairs (x, y) such that the path from x to y has exactly length 2^k, where k is the number of rounds of the TC computation. Table Q is initialized with all edges having weights greater or equal than β_i and length $2^0 = 1$ (Line 2). The table TC is used as the suffix table and for storing all TC edges at the end of the iterations.

The algorithm works as follows. Both tables Q and TC are initialized by graph *Graph*

[1] Since alignment *Graph* is a multistage graph, all paths between two nodes are of the same length.

Algorithm 2 Smart pivot graph algorithm

```
1:  Graph(x,y,s) ← Topics(x,p), Topics(y,p+1), Sim(x,y,s).
2:  Q(x,y,1,i,0) ← Graph(x,y,s), Beta(b,i), s>=b.
3:  TC(x,y,1,i,0) ← Graph(x,y,s), Beta(b,i), s>=b.
4:  Q(x,z,d1+d2,i,k) ← Q(x,y,d1,i,k-1), Q(y,z,d2,i,k-1).
5:  TC(x,y,d,i,k) ← TC(x,y,d,i,k-1).
6:  TC(x,y,d,i,k) ← Q(x,y,d,i,k).
7:  TC(x,y,d1+d2,i,k) ← Q(x,z,d1,i,k), TC(z,y,d2,i,k).
8:  Future(t,t,y,s,s,1,i) ← Graph(t,y,s), Beta(b,i), s>=b.
9:  Future(t,x,y,rs,ps,d+1,i) ← TC(t,x,d,i), Graph(x,y,rs), Sim(t,y,ps).
10: Past(t,x,t,s,s,1,i) ← Graph(x,t,s), Beta(b,i), s>=b.
11: Past(t,x,y,rs,ps,d+1,i) ← TC(y,t,d,i), Graph(x,y,rs), Sim(x,t,ps).
```

containing all edges (x, y) where $sim(x, y) \geq \beta_i$. Q serves as the prefix table and TC serves as the suffix table. Then, after each iteration k of rules Line 4 to Line 7, table Q holds all pairs of nodes connected by paths of length 2^k (rule Line 4) and table TC holds all pairs of nodes connected by paths of length at most $2^{k+1} - 1$ (rules Line 5 to Line 7). For example, after the first iteration ($k = 1$), table $Q(x, z, d1 + d2, i, 1)$ holds all paths of length $2^1 = 2$ and table $TC(x, y, d, i, k)$ holds all paths of length 1, 2 and 3. After the second iteration ($k = 2$), table $Q(x, z, d1 + d2, i, 2)$ holds all paths of length $2^k = 4$ and table $TC(x, y, d, i, k)$ holds all paths of length 1 to 7.

5.1.3 Incremental Join-based TC Computation

Algorithm 1 and Algorithm 2 compute the TC table for each β_i independently from scratch which means that certain transitive closure edges are computed several times for different beta values. The incremental approach takes advantage of the monotonicity of the pivot graphs where $TC_{\beta_{i+1}} \subseteq TC_{\beta_i}$ for all $0 \leq i < n$.

The first three rules of the Datalog program Algorithm 3 are almost identical to the first three rules of Algorithm 1. The main difference is that it starts with the TC computation for the highest threshold β_n. Then for β_i where $i < n$, the algorithm will reuse $TC_{\beta_{i+1}}$ to compute TC_{β_i} over new edges with similarity in $[\beta_i, \beta_{i+1}[$. For this, the TC computation steps on lines 2 and 3 of the baseline algorithm (Algorithm 1) are replaced by rules 4 and 5. Rule 4 finds new TC edges which are reachable from $TC_{\beta_{i+1}}$, whereas rule 5 returns the new TC edges which reach the edges in $TC_{\beta_{i+1}}$. The past and future are then obtained by the same rules as in Algorithm 1.

Algorithm 3 Incemental pivot graph algorithm

```
Graph(x,y,s) ← Topics(x,p), Topics(y,p+1), Sim(x,y,s).
TC(x,y,1,n) ← Graph(x,y,s), Beta(b,n), s>=b.
TC(x,z,d+1,n) ← TC(x,y,d,n), Graph(y,z,s), Beta(b,n), s>=b.
TC(x,z,d+1,i) ← TC(x,y,d,i+1), Graph(y,z,s), Beta(a,i), Beta(b,i+1), a <= s < b.
TC(x,z,d+1,i) ← Graph(x,y,s), TC(y,z,d,i+1), Beta(a,i), Beta(b,i+1), a <= s < b.
Future(t,t,y,s,s,1,i) ← Graph(t,y,s), Beta(b,i), s>=b.
Future(t,x,y,rs,ps,d+1,i) ← TC(t,x,d,i), Graph(x,y,rs), Sim(t,y,ps).
Past(t,x,t,s,s,1,i) ← Graph(x,t,s), Beta(b,i), s>=b.
Past(t,x,y,rs,ps,d+1,i) ← TC(y,t,d,i), Graph(x,y,rs), Sim(x,t,ps).
```

Cost Estimation We can estimate the cost for computing table *TC* by the total number of joins executed during the evaluation. For Algorithm 1, suppose that we have p periods. Then the computation cost of TC graphs corresponds to $p-1$ joins for each beta theshold β_1 and we obtain a constant cost of $n \times (p-1)$ where n is the number of thresholds. For Algorithm 2, the Smart TC requires $k = \lceil log_2(p-1) \rceil$ joins to terminate for each β_i. For Algorithm 3, the cost of the TC computation for each threshold only depends on *the maximal length of new paths* extending paths already generated for higher thresholds. This length is always smaller or equal to $p-1$ which guarantees that Algorithm 3 produces a lower or an equal number of joins than Algorithm 1.

Table Size Estimation Both algorithms (baseline and incremental) compute the same tables with the same number of distinct tuples. We can estimate the size of each table computed by both algorithms as follows. Let p be the number of periods, t be the number of topics *per period*, k be the maximal outdegree and indegree in *Graph* and b be the number of β_i thresholds. *Graph* is a (directed acyclic) multistage graph where each topic (except the leaves) have maximally k child links and the size (number of edges) of *Graph* is bound by $|Graph| \leq k * t * (p-1)$. The size of the transitive closure *TC* is bound by $|TC| \leq b * k * t * p * (p-1)/2$ (b times the size of the transitive closure of *Graph*). Finally, both tables *Future* and *Past*, contain for each tuple (x, y, d, i) in the transitive closure *TC* (x and y are connected by a path of length d), at most k tuples $Future(x, y, z, \_, \_, \_)$ and at most k tuples $Past(y, x, z, \_, \_, \_)$. Therefore, the size of *Future* and *Past* is bound by k times the size of the transitive closure: $|Future| \leq b * k^2 * t * p * (p-1)/2$. As our experiments show, even for small β-thresholds ($\beta = 0.2$) the maximum indegree and outdegree of a topics is smaller than $k = 10$ and we generally assume about $t = 100$ topics over $p = 20$ periods. Then, for $b = 10$ the size of the transitive closure is $|TC| \leq 10 * 10 * 100 * 10 * 19 = 1.9 * 10^6$ edges and the size of

$|Future| \leq 1.9 * 10^7$ edges. These numbers are much smaller in practice (see Section 5.3) and current big data frameworks can easily manage graphs of this size.

5.2 BSP-based Pivot Graph Computation

The join-based TC computation algorithms are mainly based on the efficient distributed map-reduce algebra of Spark SQL. We will show in Sections 6.1.1 to 6.1.3 how to generate pivot graphs, topic labels and graph metrics (evolution, convergence and split degrees) and how to evaluate pivot query expressions (Section 6.1.4) by simple non recursive SQL queries over the generated relational tables.

However, as will be shown by our experiments (Section 5.3.1), the transitive closure computation generates many join query tasks with an important task scheduling overhead. Therefore, we also implemented an alternative pivot graph materialization strategy based on GraphX [32], an expressive graph-processing framework. GraphX provides the Pregel [29] operator which implements the Bulk Synchronous Parallel (BSP) [96] messaging abstraction. This operator can naturally be used for efficiently computing "node-centric graph aggregations" like PageRank [101], Vertex Centrality [102] or single-source shortest path[2]. It is therefore a natural candidate for computing pivot graphs, pivot topic labels and graph metrics.

5.2.1 The Pregel operator

Pregel using Bulk Synchronous Parallel Paradigm, introduced by Google [29], is a message-passing system for processing large-scale graphs, such as web graphs and various social networks, which is expressive and easy to program. The Pregel operator of GraphX is executed in a series of iterations called super-steps, where vertices receive the sum of their inbound messages from the previous super-step, compute a new value for the vertex property, and then send messages to neighboring vertices in the next super-step. Messages are computed in parallel as a function of the edge triplet and the message computation has access to both the source and destination vertex attributes. Vertices that do not receive a message are skipped within a super step. The Pregel operator terminates iteration and returns the final graph when there are no messages remaining or the maximum number of iterations is achieved.

[2] https://en.wikipedia.org/wiki/Shortest_path_problem

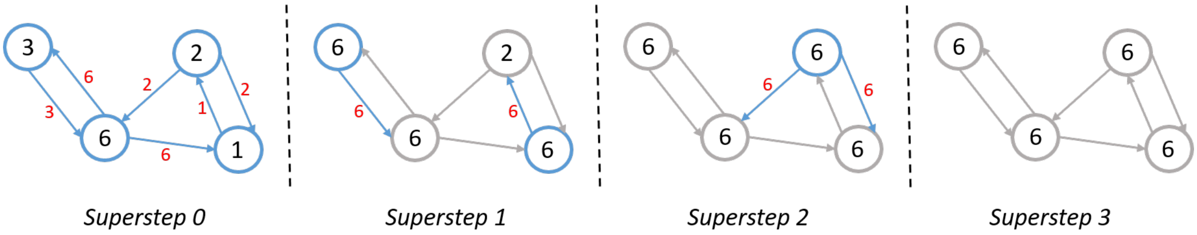


Figure 5.2: Find the maximum value in a graph. Shaded vertices will not send messages.

Example 19 *Figure 5.2 explains how Pregel works by showing a sequence of supersteps to make all vertices to have the maximum value of a vertex in a strongly connected graph where every vertex is reachable from every other vertex. In superstep* 0*, all vertices are active and send their values to their neighbors. Then in each subsequent superstep, messages received by each vertex are combined, and the largest value is found. If this value is larger than the vertex's value, then the vertex's value is updated. If none of the messages are of higher value than a given vertex, it deactivates itself. Inactive vertices will stop sending messages, but they can still receive messages and apply information in the messages received. When no further vertices change their values in a superstep, the algorithm completes. Then the largest value has been propagated to every vertex in the graph.*

5.2.2 The Pregel operator in GraphX

For illustrating the implementation of the Pregel Operator in GraphX, we present a distributed implementation for computing the liveliness of all future pivot topics.

Listing 5.1: Liveliness computation of future pivot topics by GraphX Pregel

```
import org.apache.spark.graphx._
// Initialize the graph such that each vertex has 0 as its liveliness
val initialGraph:Graph[Int, Double] = ...
// Start Pregel by sending the initial message containing 0 as the liveliness
val futureLiveliness = initialGraph.pregel(initialMsg = 0)(
  (id, oldV, newV) => math.max(oldV, newV), // Vertex program action
  triplet => { // Send message
    if (triplet.srcAttr < triplet.dstAttr + 1) {
      Iterator((triplet.srcId, triplet.dstAttr + 1))
    } else Iterator.empty
  },
  (a, b) => math.max(a, b) // Merge message
13 )
```

Listing 5.1 shows the liveliness computation of future pivot graphs expressed by GraphX Pregel operator in Scala. The future liveliness of a pivot topic corresponds to the graph diameter (the length of the longest path from the topic to a future descendant topic). Since future pivot graphs are directed and acyclic, children vertices have smaller future liveliness than their parent vertices. Therefore, this computation is a bottom-up process. More precisely, children vertices in each super step in the up coming loop will send message to their parent vertices if their liveliness is no smaller than their parents' liveliness. The message to be sent is the children vertex's liveliness plus 1.

The initial graph on line 3 is a graph with topic nodes connected by alignment edges. The Pregel operator `initialGraph.pregel` contains two argument lists and returns a new graph $futureLiveliness$ where each vertex has a liveliness value corresponding to the diameter of its subgraph. The first argument list contains the configuration parameters including the initial message, the maximum number of iterations (by default the maximum positive value for a 32-bit signed binary integer), and the edge direction in which to send messages (by default along out edges). The initial message is to start the computation. This message is passed to all the vertices to do the first iteration. In this example, we define the initial message $initialMsg$ by 0 liveliness (line 5).

The second argument list contains three user defined functions after receiving a message (Vertex program, line 6), for sending a message (Send message, lines 7 to 11) and combining received messages (Merge message, line 12). The Vertex program takes action after each received message to set the vertex attribute, *i.e.*, liveliness, to the maximal value of its old value and the new received value. This action corresponds to the second user defined function (lines 7 to 11). During each super step, each graph edge triplet $tiplet(srcAttr, attr, dstAttr)$ examines whether a message should be sent from the destination node (child) to its source (parent). When the parent's liveliness $triplet.srcAttr$ is smaller than its child vertex's $triplet.dstAttr$, a message with the child vertex's liveliness plus 1 is sent to the parent vertex (line 9). If the receiving parent vertex has connections to more than one child vertex and received more than one messages, Merge message chooses the maximal value (line 12).

Algorithm 4 Incremental Pregel for Future Pivot Graph Generation

```
Input: $\mathcal{G}$ : Graph[V, E] where V contains $vid$ and E contains $(src, sim, dst)$ tuples
        $g$ : Graph[V, E] where V contains $(vid, (live(v), subgraph(v)))$ and E contains
            $(src, sim, dst)$ tuples
Output: $g$
for $i = n...0$ do
    // initialize pivot graph
    if i=n then
        $g = Graph(g.V, g.E'), g.E' \in g.E, \forall e \in g.E', e.sim \geqslant \beta_n$
    else
        $g = Graph(g.V, \mathcal{G}.E' \bigcup g.E), \mathcal{G}.E' \in \mathcal{G}.E, \forall e \in \mathcal{G}.E', e.sim \in [\beta_i, \beta_{i+1}[$
    end if
    // Pregel superstep loop
    while $\exists e$, $e \in g.E$, $live(e.dst) \geqslant live(e.src)$ do
        // send messages
        do the following process in parallel
            send message $(live(e.dst) + 1, (subgraph(e.dst).add(e)))$ from $e.dst$ to $e.src$
        // receive messages and update nodes' attributes
        do in parallel when $e.src$ has received message $(dstlive, dstedges)$ from $e.dst$
            $live(e.src) = max(live(e.src), dstlive)$
            $subgraph(e.src) = subgraph(e.src) \bigcup (dstedges)$
    end while
end for
```

5.2.3 Future Pivot Graph Generation with Pregel

The Algorithm 4 describes the generation of future pivot graphs using the GraphX Pregel operator. The generation applies the idea of the incremental computation approach as described in Section 5.1.3 to avoid redundant TC edge computations by reusing TC edges computed for higher beta thresholds. The input of the algorithm are a phylomemetic graph $\mathcal{G}$ and a pivot graph g where each topic vertex v stores its liveliness value $live(v)$ and its future subgraph edges $subgraph(v)$. By default, the attributes $live(v)$ and $subgraph(v)$ of all nodes are initialized by 0 and the empty set respectively.

The computation starts from the highest β value. At the first β iteration ($\beta = \beta_n$), each edge in the initial graph which satisfies the condition $sim \geqslant \beta_n$ will be used to generate pivot graphs (line 8). We use the topic liveliness values to define the super-step condition to stop the computation process. The algorithm converges when, for each alignment edge e, the liveliness of the source vertex *e.src* (parent topic) is greater than the liveliness of its destination vertex *e.dst* (child topic), *i.e.*, $live(e.dst) < live(e.src)$. Then at each super-step (line 13 to line 21), every destination vertex *e.dst* which satisfies the super-step condition $live(e.dst) >= live(e.src)$ will send messages to its parent node $live(e.src)$ (line 14 to line 16). Each message contains a new liveliness value by adding 1 to the topic's previous liveliness value as well as the subgraph alignment edges of the vertex. Each (parent) node which receives these messages will reduce the received values to one value and update its attributes (line 18 to line 20). After the super-step loop, each vertex contains the incremented maximum liveliness value and the union of all subgraph edges of its children.

For the remaining beta values (line 10), the computation starts from the all nodes and edges obtained by the previous iterations (for higher β value) and the subset of edges which satisfy the condition that $sim \in [\beta_i, \beta_{i+1}[$. It is easy to see that this initialization reuses the liveliness and the subgraph edge sets obtained by the previous β iterations.

5.3 Experiments

We conducted various experiments to study the performance of the proposed different pivot graph computation models. We study join-based computation models by first comparing the performance of the baseline approach and the incremental approach presented in Section 5.1.1 and Section 5.1.3. Then the comparison of the three relational computation models demonstrates that the incremental approach performs better by computing fewer number of TC joins and fewer number of TC edges. Then the parallelization and the scalability of the pivot graph generation process are discussed in Section 5.3.2. Finally in Section 5.3.3, we compare the execution time of the incremental join-based model with the incremental BSP-based model which reveals a better performance.

5.3.1 Comparison of Relational TC Computations

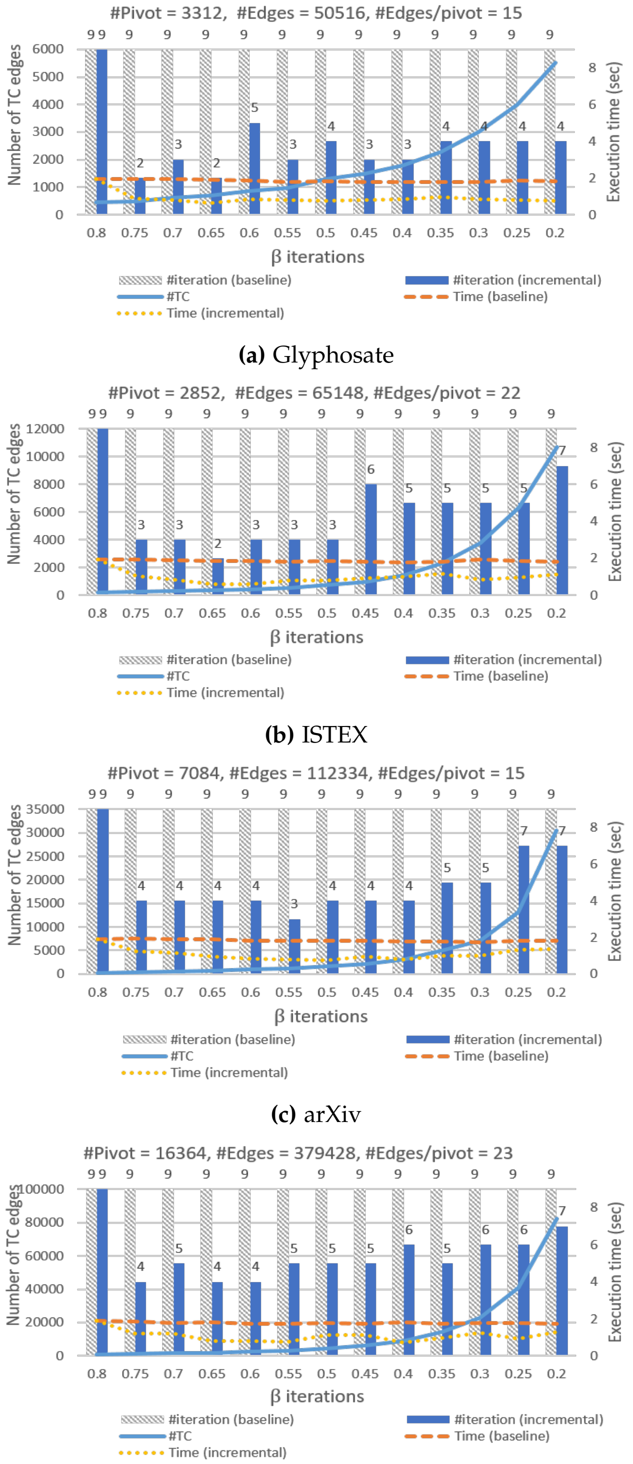
TC Graph Computation Performance

The generation of pivot graphs is the main computation step of our workflow including many join operations which are necessary to compute the transitive closure (TC graph) of pivot graphs for different β values.

We compare the performance of two algorithms presented in Section 5.1.1 and Section 5.1.3. The baseline algorithm computes independently for each β threshold, the pivot graphs of all corresponding pivot topics from scratch whereas the incremental approach reuses the result of previous β-iterations to compute the pivot graphs of new topics for a lower β value.

Figure 5.3 shows for each corpus, the transitive closure (TC) computation time of the baseline algorithm (orange line) and the incremental algorithm (yellow line). These experiments have been applied on the real-world data sets described in Table 3.1. The β values span from 0.2 to 0.8. The baseline algorithm (gray bars) applies for each β threshold 9 iterations (1 join per iteration) for generating the TC edges over all 10 periods, whereas the number of joins in the incremental version (blue bars) varies between 2 and 7 iterations, except for the first β value. It can be seen that the execution time of the incremental algorithm (yellow dotted dashed line) is reduced by about 50% compared to the execution time of the baseline algorithm (orange dashed line). We also can observe that, even if the size of the joined table increases exponentially (blue solid line), the execution time mainly depends on the number of iterations (joins) whereas the cost of each individual join remains constant.

To better illustrate the benefit of the incremental evaluation, we have conducted 3 groups of experiments for each corpus with β intervals ranging from 0.2 to 0.8 of different granularity for the purpose of having more β values. The results are shown in Figure 5.4. For the first group of experiments, the interval step is 0.1 (7 β values), for the second group 0.05 (13 β values) and for the third group 0.02 (31 β values). The bar chart represents the total



(a) Glyphosate

(b) ISTEX

(c) arXiv

(d) Wiley

Figure 5.3: Correlation between the execution time and the number of iterations of transitive closure computation for each β iteration

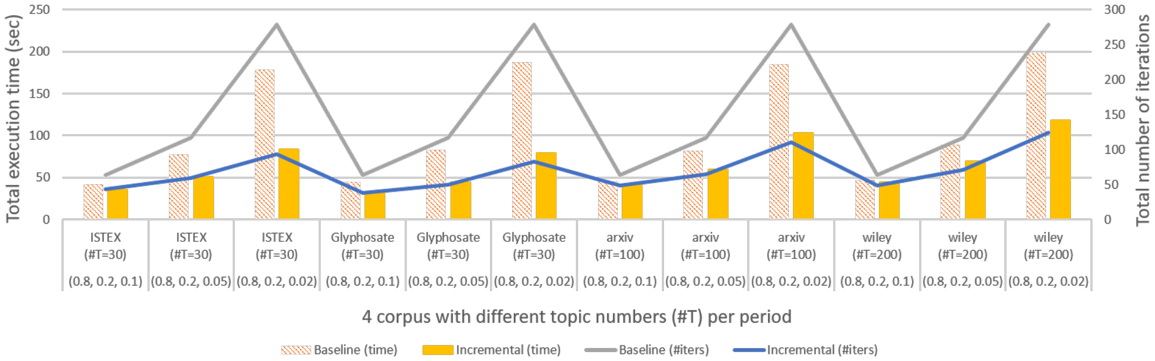


Figure 5.4: Comparison between the baseline and the incremental TC computation

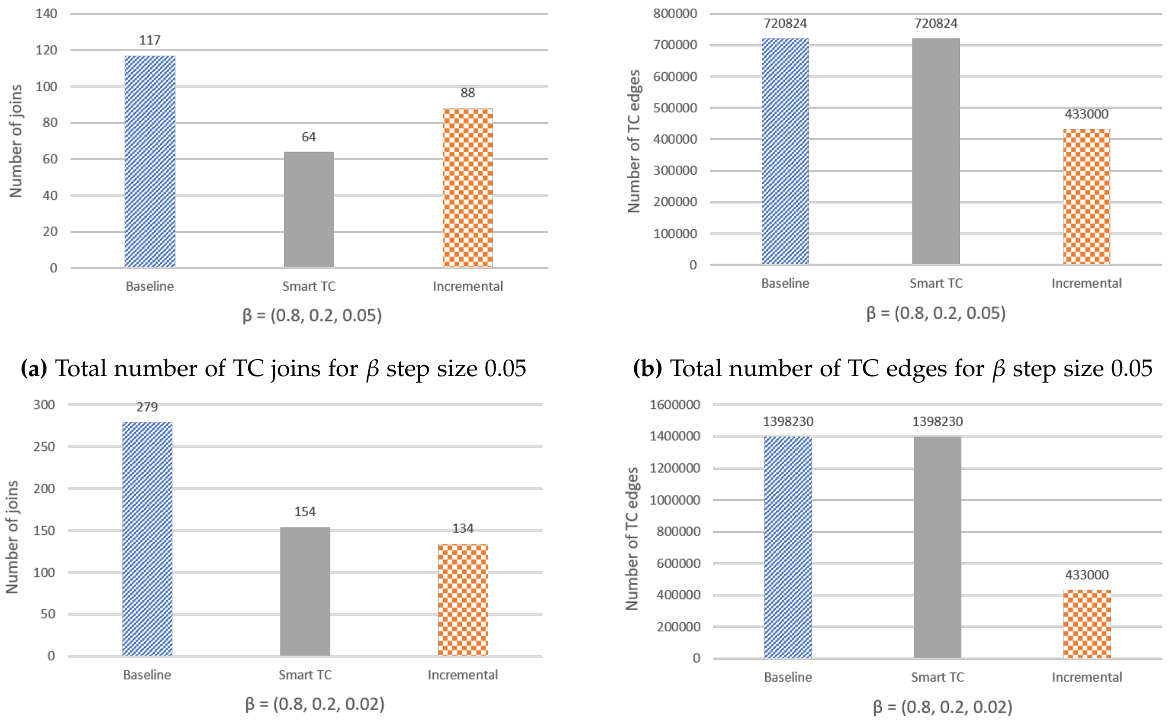
TC computation time and the line chart represents the total number of TC iterations. It is easy to see that the computation time mainly depends on the number of iterations and the incremental algorithm outperforms the baseline solution.

Computation Cost and Data Size Comparisons

Figure 5.3 demonstrates that the execution time of TC computation is proportional to the number of iterations, *i.e.*, the number of TC joins. Here we compare with the smart approach the total number of TC joins of the two approaches evaluated in the previous experiment. And another way to compare these algorithms is to compare the total number of TC edges (including redundant edges) produced by all algorithms.

We conducted two groups of experiments where we used a 20-period phylomemetic graph generated from the arXiv[3] corpus, chose (0.2, 0.8) as the β interval and set the β step to 0.05 and 0.02 respectively. The results of these two groups of experiments are presented in Figure 5.5 where each row of 2 subgraphs corresponds to a group of experiment. On the first row, we set β step to 0.05 ($(0.8 - 0.2)/0.05 + 1 = 13$ β values). On the second row, the β step is 0.02 ($(0.8 - 0.2)/0.02 + 1 = 31$ β values). When comparing Figure 5.5a with Figure 5.5c, it can be easily seen that the smart approach benefits of the reduced number of joins in a logarithmic manner for each β iteration to outperform the baseline algorithm which results in the total number of joins reduced by half. The incremental approach also benefits of the reduced number of joins compared with the baseline approach for the reason that the number of joins depends on the length of the longest path of new edges extending paths already

[3] https://arxiv.org/



(a) Total number of TC joins for β step size 0.05

(b) Total number of TC edges for β step size 0.05

(c) Total number of TC joins for β step size 0.02

(d) Total number of TC edges for β step size 0.02

Figure 5.5: Comparison of the total number of TC joins and the total number of TC edges by the 3 join-based approaches over arXiv corpus with 20 periods in total

generated for higher thresholds. Therefore, a smaller β step leads to new shorter edges at each β iteration. For this reason, we can explain that in Figure 5.5a, the smart approach outperforms other approaches whereas in Figure 5.5c, the incremental approach reduces the number of joins even more than the smart approach.

Except the reduced number of TC joins, the incremental method also benefits of the reduced size of joined data (TC edges) due to the reuse of TC table of β_{i+1} value at the β_i iteration. As can be seen, for β step 0.05, the total number of TC edges computed by the incremental approach which is 433 000 only accounts of 60% of the other approaches. The baseline algorithm and the smart algorithm generate almost 40% of redundant edges compared to the increment algorithm. For β step 0.02, the baseline method and the smart method compute 1 398 230 TC edges which is three times larger than the number of TC edges computed by the incremental approach. The baseline approach and the smart approach generate almost 70% of redundant edges compared to the increment algorithm. The smaller the β step is, the more efficiently the incremental approach performs.

5.3.2 Parallelization and Scalability

In this section, we first talk about the performance evaluation of the entire pivot graph generation process computed by the incremental join-based approach over the four real-world datasets described in Table 3.1. We found that the incremental algorithm allows to extract pivot graphs from real-word datasets on a local machine. For demonstrating the scalability of the pivot graph generation, we then evaluate the TC computation performance on a cluster of nodes with much larger (synthetic) datasets.

Small real-world pivot topic graphs

Figure 5.6 illustrates the execution time for computing pivot graphs, topic labels and graph metrics for four different real-world document archives presented in Table 3.1. We measure the performance obtained for different numbers of CPU cores on a local machine. The β values span from 0.2 to 0.8 with a step of 0.05 (13 values). As can be seen, the pivot graph computation time dominates the execution time of these three steps. The execution time for topic labeling accounts for half of the execution time for pivot graphs and the execution time for graph metrics remains constant and takes about 10 seconds. Observe that the best performance has been achieved with a single CPU core on local machine. This can be

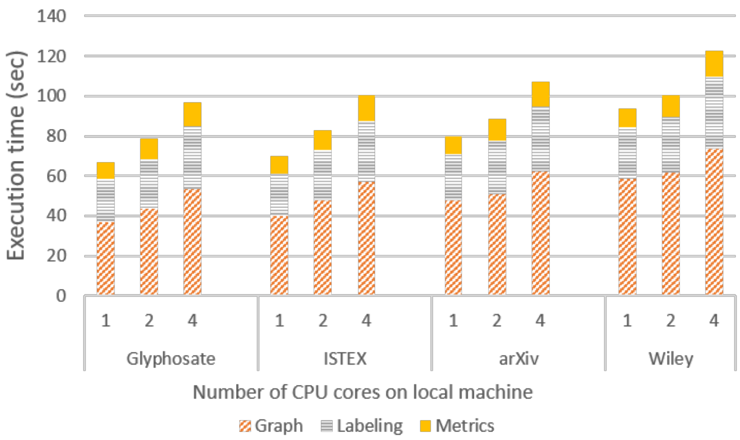


Figure 5.6: Pivot Graph Computation : CPU cores vs. execution time

explained by an increase of the task deployment and coordination overhead which dominates the query execution time. More precisely, increasing parallelism on a single machine will also increase stages which results in more operations. Furthermore, cost of supervising small tasks is significant as well. Finally, the execution of our algorithms mainly generate hundreds of “short” query tasks, which can efficiently be serialized on a single CPU.

Large Synthetic Pivot Graphs

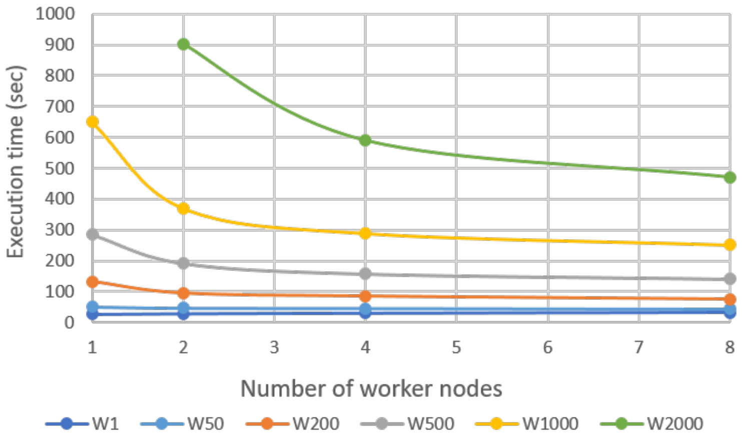


Figure 5.7: Performance evaluation on the cluster with synthetic datasets

For illustrating the pivot graph computation performance on much larger graphs, we generated several synthetic datasets (topic graphs) by duplicating the Wiley topic graph.

The results of these experiments are shown in Figure 5.7. Each line corresponds to the TC graph computation time of a synthetic topic graph *Wx* obtained by generating *x* copies of the initial Wiley graph containing 2000 topics and 360 000 alignment edges. Thus *W1* contains the topic graph as mentioned in Figure 5.3 whereas *W2000* is the largest synthetic dataset with 4 million topics and 720 million alignment edges. This last graph generates 32 million pivot graphs with 7, 6 billion edges with a total size of 30 GB on disk.

The number of physical worker nodes ranges from 1 to 8 and each worker node contains one Spark executor configured with 4 CPU cores and 40 GB memory. Figure 5.7 illustrates the TC computation time for the synthetic topic graphs $W1$, $W50$ $W200$, $W500$, $W1000$ and $W2000$. The edges of all graphs are distributed randomly on all worker nodes. As we can see, the performance benefit for $W1$ and $W50$ by increasing the number of worker nodes is very low. As already mentioned, these two datasets cannot benefit from the cluster since the cost of each task is small compared to the Spark task scheduling and data shuffling overhead. For larger graphs, it is possible to achieve a $2X$ speedup by increasing the number of worker nodes. We also see that a single node is not able to process $W2000$. The relative benefit decreases with the number of nodes. This is mainly due to the removal of duplicate edges during the TC computation, which needs data shuffling between nodes and adds communication cost.

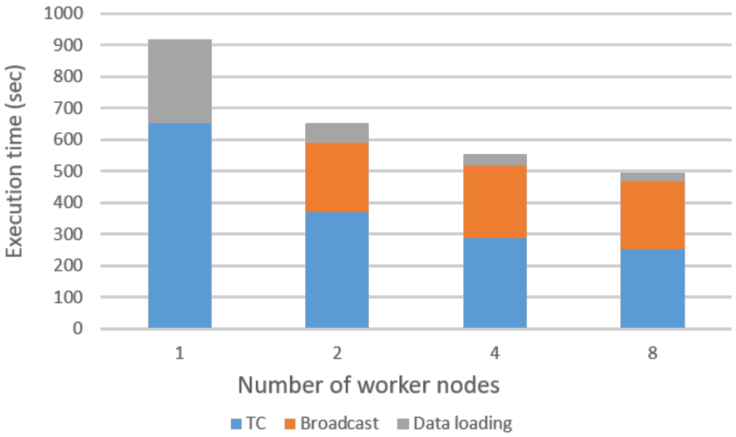


Figure 5.8: Execution time of different steps to compute TC for W1000 on the cluster VS. number of worker nodes

Figure 5.8 shows the total execution time according to the different TC computation steps applied on $W1000$ with 1 up to 8 worker nodes. The blue bars correspond to the iterative TC compution costs shown in Figure 5.7 (yellow line). For example, the TC computation time on one node in Figure 5.8 is 650 seconds which corresponds to the point of the yellow line for one node in Figure 5.7. We apply the distributed broadcast join algorithm implemented in Spark SQL[4]. The cost of broadcasting the TC table to be joined at each iteration is shown in orange except for the single node architecture, which can apply a simple centralized join operator. The total broadcast time mainly depends on the total data size exchanged between all nodes and is almost constant for 2, 4 and 8 nodes. Loading the data from the distributed HDFS[5] file system into memory is improved up to 10 times (from 268 seconds to 26 seconds) by using 8 nodes instead of 1 (grey bar on the top).

5.3.3 Join-based versus Pregel-based TC Computation

In this section we compare the performance of the relational and graph-based pivot graph computations presented in Section 5.1 and Section 5.2. Both computations are distributed. The relational solution uses a distributed algorithm for joining tables distributed on different nodes and the graph based solution uses the distributed Bulk-Synchronuous Protocol for exchanging messages between nodes. The experiment is carried out on the single machine with a hyperthreaded 3.1 GHz Intel Core i7-7920HQ processor as described in Section 3.2.2. We have implemented all the three proposed join-based pivot graph computation models and in an EPIQUE workflow, the choice of the model depends on the user. Since the incremental approach has the best performance among our relational implementations, it is used to compare with the incremental implementation of GraphX Pregel. The results are shown in Figure 5.9. We conducted four groups of experiments for two different β interval granularities and phylomemies over 10 and 20 periods.

In Figure 5.9, we can see that the incremental GraphX Pregel method always outperforms the relational incremental approach regardless of the granularity of β intervals. However for phylomemies with 20 periods, the performance gain obtained by the incremental GraphX Pregel diminishes. This can be explained by the increased size of the messages exchanged between the topic nodes (at each superstep, all topic nodes communicate all their descendants to their parent node).

Figure 5.10 gives a deeper insight about the performance of subgraph computation for each β iteration of the incremental Pregel method. As can be seen in Figure 5.10a, the execution time is almost constant for $\beta \in [0.8, 0.6]$ and increases afterwards for the smaller β values in $[0.55, 0.2]$ decreases. This can be explained by the increased number and increased size

[4] https://mungingdata.com/apache-spark/broadcast-joins/

[5] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

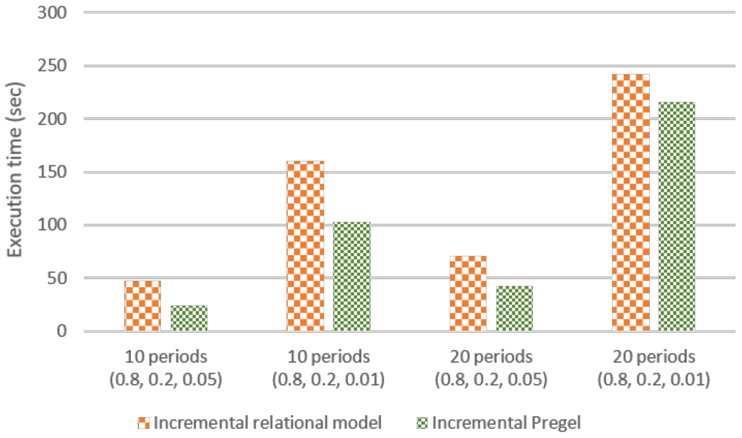
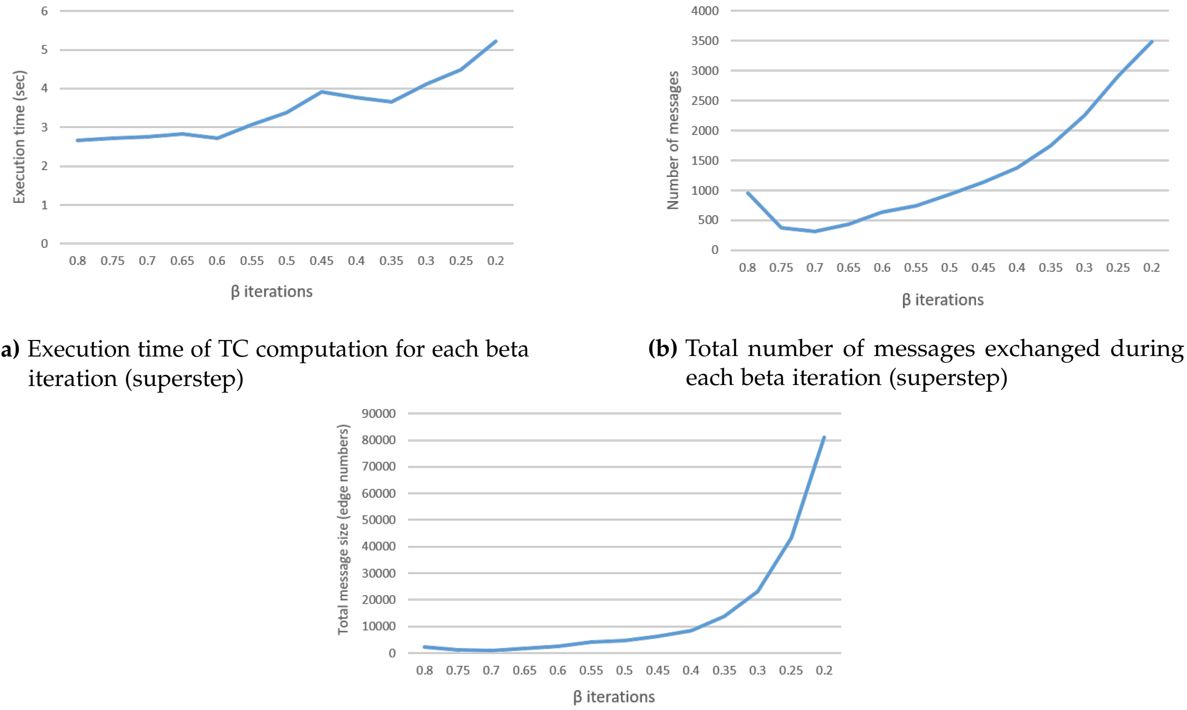


Figure 5.9: Performance comparison for subgraph computation between the relational incremental approach and the incremental Pregel approach over arXiv corpus



(a) Execution time of TC computation for each beta iteration (superstep)

(b) Total number of messages exchanged during each beta iteration (superstep)

(c) Total size of all messages (graph edge number) exchanged during each beta iteration (superstep)

Figure 5.10: Illustration of the impacts on the performance of incremental Pregel tested over arXiv corpus with 20 periods in total

of messages being sent during the supersteps as shown in Figure 5.10b and Figure 5.10c. As shown in Figure 5.10b, the number of messages decreases after the first β iterations and begins to increase at $\beta = 0.7$. The total size of the messages (number of subgraph edges) $\beta \in [0.8, 0.6]$ iterations is almost constant as shown in Figure 5.10c. Then, from $\beta = 0.6$ upwards, the total size of messages for each β value starts to increase whereas the execution time grows from the same point.

5.3.4 Conclusion

Our previous experiments demonstrate that the execution time of join-based TC computation mainly depends on the number of join operations. The incremental algorithm outperforms the baseline algorithm by both reducing the number of joins and the number of TC edges to be computed. Compared to the smart algorithm, the incremental algorithm reduces more TC joins when β step is smaller and generates fewer TC edges regardless of β step. The experiments also reveal that our incremental join-based algorithm dominates the execution time of the entire pivot graph generation process and allows to efficiently extract pivot graphs from real-world datasets on a single machine. The performance experimental evaluation on cluster shows that the distributed computation on several nodes increases the execution cost due to additional data shuffling and task management overhead. Whereas this observation questions the choice of using Spark which is mainly designed for parallel data-intensive computations on clusters. In particular, Spark uses the Hadoop file system (HDFS) [103], which is designed to efficiently manage large persistent DataFrames and to reduce distributed data access/storage overhead compared to a standard file system solution. Nevertheless, the previous experiments do not take account of all workflow steps including the document storage, preprocessing (stop-word removal, stemming, term extraction) steps which can benefit of a distributed Spark architecture.

6 Implementation and Prototype

This chapter presents the implementation of the topic evolution model in the context of the EPIQUE project. Our implemented workflow is able to handle any scientific document corpus where each document has a publication date and some text contents (title, abstract, keywords, etc.). I will first explain our implementation of pivot graph generation and pivot query evaluation in the following section. Section 6.2 then introduces our prototype implementation including the architecture of the EPIQUE web application as well as a scenario for pivot graph generation, exploration and analysis.

Contents

6.1 Pivot Graph Generation and Query Evaluation

This section explains in detail the implementation of the entire pivot graph generation workflow including the subgraph generation step with the incremental approach (Section 6.1.1). This step mainly generates two Spark Dataframes [104] *Past* and *Future* which contain all information necessary for the following steps. In particular, I show how to implement the topic labeling step (Section 6.1.2) and the graph metrics computation step (Section 6.1.3) by using Spark SQL [104]. Then I illustrate in Section 6.1.4 how pivot queries can be evaluated using standard SQL queries.

6.1.1 Pivot Graph Computation with Spark

Algorithm 5 shows our pivot graph computation implementation using the Spark DataFrame [104] model and algebra. A Spark DataFrame is equivalent to a relational database table and can be transformed into a new DataFrame through relational operations including projection (*select*), filter (*where*), *join*, and aggregations (*groupBy*). Algebraic expressions are built by concatenating the relational operators starting from the input DataFrame. For example, the expression in line 5, applies the following relational selection on DataFrame *Sim*:

$$Graph = \sigma_{sim \geqslant \beta_i \wedge t_1.period+1=t_2.period}(Sim)$$

A projection of DataFrame *Graph* on the three attributes t_1, t_2, l is shown in line 10 (attribute l is intialized with value 1). Finally, line 15 defines a relational join between the two DataFrames *newGraph* and *oldTC* followed by a projection where attribute *newGraph.d* + 1 is incremented by one:

$$newTCEdges = \pi_{newGraph.t_1, oldTC.t_2, newGraph.d+1}(newGraph \bowtie_{newGraph.t_2=oldTC.t_1} oldTC)$$

Algorithm 5 computes and stores for each alignment threshold β_i ($0 \leq i \leq n$) two tables *Future* and *Past* as defined in Section 5.1. The pivot graph computation relies on an efficient transitive closure algorithm to connect all the reachable topics starting from any pivot topic. This computation may be expensive for large and highly connected graphs. The main idea of the proposed implementation is to benefit from the inclusion property $\mathcal{G}^*(t, \beta_{i+1}) \subseteq \mathcal{G}^*(t, \beta_i)$ ($0 \leq i < n$) mentioned in Section 5.1. This property implies that, for decreasing i, the transitive closure TC_{i+1} obtained for β_{i+1} is included in the transitive closure of TC_i for β_i and therefore gives the opportunity to reuse the transitive closure results among the successive iterations.

The iterative computation starts on line 3 from the highest value β_n. The first step (lines 5) defines the complete *Graph* for β_i containing all alignment edges where $sim \geq \beta_i$. The

Algorithm 5 Pivot graph computation using the Spark data model and algebra

```
Input: topic similarity DataFrame $Sim(t_1, t_2, sim)$
Output: a pair $(Future, Past)$ of future and past pivot graph DataFrames for each beta value $\beta_i$.
for $i = n...0$ do
    // Graph($t_1$, $t_2$, sim) : topic alignment graph for $\beta_i$ (rule 1 in Algorithm 1)
    Graph = Sim.where($sim \geqslant \beta_i \wedge t_1.period + 1 = t_2.period$)
    // Transitive closure computation (rules 3 and 4 in Algorithm 1)
    if i=n then
        oldTC = $\emptyset$                                        ▷ oldTC is empty for $\beta_n$
        // newGraph($t_1$, $t_2$, d) : new alignment edges (d=1) where sim $\geq \beta_n$
        newGraph = Graph.select($t_1$,$t_2$,1 as d)
    else
        // newGraph($t_1$, $t_2$, d) : new alignment edges (d=1) where sim $\in [\beta_i, \beta_{i+1}[$
        newGraph = Graph.where(sim < $\beta_{i+1}$).select($t_1$,$t_2$,1 as d)
        // newTCEdges($t_1$, $t_2$, d) : new alignment edges (d>1) between newGraph and oldTC
        newTCEdges = newGraph.join(oldTC, newGraph.$t_2$ = oldTC.$t_1$)
                .select(newGraph.$t_1$, oldTC.$t_2$, newGraph.d + 1 as d)
        newGraph = newGraph.union(newTCEdges)    ▷ add newTCEdges to newGraph
    end if
    deltaTC = oldTC.union(newGraph)              ▷ deltaTC($t_1$, $t_2$, d): oldTC + newGraph
    // *Semi-naive* evaluation (rule 3 in Algorithm 1)
    newTC = deltaTC                              ▷ newTC($t_1$, $t_2$, d) : trans. closure of deltaTC
    while deltaTC $\neq \emptyset$ do
        deltaTC = deltaTC.join(newGraph, deltaTC.$t_2$ = newGraph.$t_1$)
                .select(deltaTC.$t_1$, newGraph.$t_2$, deltaTC.d + newGraph.d)
        newTC = newTC.union(deltaTC)
    end while
    // Generate Future(p,$t_1$, $t_2$,sim, d, psim) (rules 4 and 5 in Algorithm 1)
    Future = Graph.select($t_1$ as p, $t_1$, $t_2$, sim, 1, sim as psim)
    FutureTC = newTC.join(Graph, newTC.$t_2$ = Graph.$t_1$)
            .join(Sim, Sim.$t_1$ = newTC.$t_1$ $\wedge$ Sim.$t_2$ = Graph.$t_2$)
            .select(newTC.$t_1$ as p, Graph.$t_1$, Graph.$t_2$, Graph.sim, newTC.d+1 as d, Sim.sim as psim)
    Future = Future.union(FutureTC)
    // Generate Past(p,$t_1$, $t_2$,sim, d, psim) (rules 6 and 7 in Algorithm 1)
    Past = Graph.select($t_2$ as p, $t_1$, $t_2$, sim, 1, sim as psim)
    PastTC = newTC.join(Graph,newTC.$t_1$ = Graph.$t_2$)
            .join(Sim, Sim.$t_1$ = Graph.$t_1$ $\wedge$ Sim.$t_2$ = newTC.$t_2$)
            .select(newTC.$t_2$ as p, Graph.$t_1$, Graph.$t_2$, Graph.sim, newTC.d+1 as d, Sim.sim as  psim)
    Past = Past.union(PastTC)
    store Future and Past on disk                ▷ store all future and past pivot graphs for $\beta_i$
    oldTC=newTC
end for
```

following steps (lines 7 to 18) compute the table *newGraph* with all new direct alignment edges where $sim \in [\beta_i, \beta_{i+1}[$ and all new "transitive" alignment edges which exist between the new nodes and the previous transitive closure table *oldTC* ($oldTC = \emptyset$ for $i = n$). Observe that *newGraph* contains only edges that do not exist in *oldTC* and connects all new nodes in *newGraph* to all reachable nodes in *oldTC*. However, it still misses the new edges which might connect two nodes in *oldTC* by a new path generated by the new edges. These new edges can be obtained by computing the transitive closure on table *deltaTC*, which is initialized with all new edges in *newGraph* and the old TC edges in *oldTC* (line 19). We then have the following definition of *deltaTC* (the join predicates are ignored for simplification):

$$deltaTC = (newGraph \bowtie oldTC) \cup oldTC \cup newGraph$$

We then compute the transitive closure *newTC* of *deltaTC* (lines 21 to 26) where each step joins *deltaTC* only with new edges in *newGraph* until reaching the fixpoint where $deltaTC = \emptyset$:

$$\begin{aligned} newTC_0 &= deltaTC \\ deltaTC_{i+1} &= deltaTC_i \bowtie newGraph \\ newTC_{i+1} &= newTC_i \cup deltaTC_{i+1} \end{aligned}$$

This strategy is similar to the semi-naive transitive closure algorithm [98] and guarantees that no TC edge is computed twice in the whole process. More precisely, the number of iterations (joins) is bound by the length of the longest path composed of new edges with $sim \in [\beta_i, \beta_{i+1}[$. This can drastically reduce the computation cost as shown in our experiments. The remaining steps (lines 27 to 38) generate the *Future* pivot graphs and *Past* pivot graphs by joining the tables *newTC*, *Graph* and *Sim* (rules 4 and 7 in Algorithm 1). Both tables are stored at the end of each iteration step.

6.1.2 Topic Label Computation

Pivot topic labels can be computed with Spark SQL [104] by defining a query over the pivot evolution Dataframes $Future(t, x, y, rs, ps, d, i)$ and $Past(t, x, y, rs, ps, d, i)$ computed by Algorithm 5 presented in Section 6.1.1.

The following query creates three temporary views *FLabels*, *PLabels* and *TmpLabels*. The first two views contain for each pivot topic (t, i) the list of future and the past terms respectively. The query uses the aggregate function *group_concat* to concatenate the terms of the future and past topics for each pivot topic. The view *TmpLabels* combines the previous two views in order to facilitate the generation of topic labels in the final query expression applying the term label predicates defined in Section 4.4. The final query on view *TmpLabels* uses three set operators *group_intersect*, *group_except* and *group_union* to compute the stable, emerging, decaying and specific terms as defined in Section 4.3. The result is stored in a new table *PivotLabels*. Notice that a given topic with a given β value has a group of fix labels (emerging, stable, decaying and specific). These labels depend on future pivot graph and past pivot graph of that topic. Thus a term can be "stable", although it does not appear anywhere in the predecessor nodes (or the successor nodes) of a certain pivot topic graph.

Listing 6.1: Topic labeling query

```
create table PivotLabels as (
 with FLabels as (
  select t, i, group_concat(y.terms) as futureTerms
    from Future
   group by t, i
 ),
 PLabels as (
  select t, i, group_concat(x.terms) as pastTerms
    from Past
   group by t, i
 ),
 TmpLabels as (
  select t, i,
         array_intersect(t.terms, futureTerms) as futureLabel,
         array_intersect(t.terms, pastTerms) as pastLabel
    from FLabels tab1, PLabels tab2
   where tab1.t = tab2.t and tab1.i = tab2.i
 )
 select t, i,
        array_intersect(futureLabel, pastLabel) as stable,
        array_except(futureLabel, pastLabel) as emerging,
        array_except(pastLabel, futureLabel) as decaying,
        array_except(t.terms, array_union(futureLabel, pastLabel)) as specific
  from TmpLabels
)
```

Example 20 *Table 6.1 shows the pivot topic* $(t51, 0.5)$ *obtained from the ISTEX corpus with its top-10 weighted terms. I will explain the topic labeling process by using the tables related to topic* $t51$ *as shown in Table 6.2. From the* $Future(t, x, y, rs, ps, d, i)$ *table and* $Past(t, x, y, rs, ps, d, i)$ *table, we can find all future topics (*$t75$ *and* $t104$*) and past topics (*$t2$*) of pivot topic* $(t51, 0.5)$ *as well as their top weighted terms, as shown in Table 6.2a and Table 6.2b. Then the fusion of all terms in future topics and in past topics generates the* $FLabels$ *table (Table 6.2c) and the* $PLabels$ *table (Table 6.2d) of pivot topic* $(t51, 0.5)$. *From the previous two tables, all terms of topic* $t51$ *that also appear in future and pas topics (shown in bold) are selected, respectively, as the* $futureLabel$ *and the* $pastLabel$ *of topic* $t51$ *(attributes in table* $TmpLabels$ *shown in Table 6.2e). Observe that term "uncertainty" is stable (appears in the past and the future), whereas "risk" and "probability" are decaying (only appear in the past) and "decisions" and "information" are emerging (appear only in the future). This final classification is displayed in Table 6.2f and obtained by the final Spark SQL expression shown in Listing 6.1. The different colors of the terms correspond to different term labels that have been explained in Figure 1.1.*

Table 6.1: Pivot topic $(t51, 0.5)$ of ISTEX corpus

Topic	Terms
51	decisions, information, risk, uncertainty, decision-making, probability, air quality, direction, assessment, complexity

6.1.3 Pivot Graph Metrics Computation

The liveliness, relative evolution degree, pivot evolution degree, split degree and convergence degree of all pivot evolution graphs can aso directly be computed by a standard SQL aggregation query. The following query computes these metrics for all future pivot evolution graphs stored in $Future(t, x, y, rs, ps, d, i)$:

Listing 6.2: Graph metrics computation query

```
create table PivotFuture as
 select t,i max(d) as liveliness,
        1−avg(rs) as revol,
        1−avg(ps) as pevol,
        count(∗)/count(distinct x) as split,
        count(∗)/count(distinct y) as conv
 from Future
 group by t,i
```

Topic	futureNode	Terms
51	75	preferences, incentives, allocation, uncertainty, decisions, transaction costs, public, asymmetric information, social welfare, geographic information systems
51	104	imports, environmental degradation, variety, stakeholders, economic activity, coastal zones, gap, asymmetric information, environmental accounting, earth

(a) Topic nodes in the future of pivot topic $(t51, 0.5)$

Topic	pastNode	Terms
51	2	uncertainty, experience, life, risk reduction, probability, trade, state, risk-aversion, environmental risks, individuals

(b) Topic nodes in the past of pivot topic $(t51, 0.5)$

Topic	futureTerms
51	preferences, incentives, allocation, **uncertainty**, **decisions**, transaction costs, public, social welfare, asymmetric **information**, geographic **information** systems, imports, environmental degradation, variety, stakeholders, economic activity, coastal zones, gap, environmental accounting, earth

(c) FLabels of pivot topic $(t51, 0.5)$

Topic	pastTerms
51	**uncertainty**, experience, life, **risk** reduction, **probability**, trade, state, **risk**-aversion, individuals, environmental **risks**

(d) PLabels of pivot topic $(t51, 0.5)$

Topic	futureLabel	pastLabel
51	decisions, information, uncertainty	risk, uncertainty, probability

(e) TmpLabels of pivot topic $(t51, 0.5)$

Topic	stable	emerging	decaying	specific
51	uncertainty	decisions, information	risk, probability	decision-making, air quality, direction, assessment, complexity

(f) PivotLabels of pivot topic $(t51, 0.5)$

Table 6.2: Tables of topic labeling process

6.1.4 Pivot Query Evaluation

The result of the previous steps are four tables, *PivotFuture*, *PivotPast*, *PivotAll* and *PivotLabels*. Based on these tables, it is possible to translate a pivot query into a standard SQL query where each pivot filter becomes a predicate in the query where clause:

Listing 6.3: Pivot query example

```
select *
  from PivotFuture topic, PivotLabels label
 where topic.t = label.t and topic.i = label.i
   and contains(label.emerging, 'deep␣learning')
   and contains(label.stable, 'database')
   and topic.liveliness > 8 and topic.split < 3
```

Example 21 *Listing 6.3 expresses the following pivot query which filters all pivot topics with emerging term "deep learning" and stable term "database", where each future topic has fewer than three child topics in average and there exist future subtopics related to the pivot topic with a distance of longer than 8 periods:*

*Q: DB.**Future**.**Emerge**("deep␣learning").**Stable**("database").**Live**(>8).**Split**(<3)*

In our prototype implementation, we defined a class *Pivot* which encapsulates all Dataframes and implements each pivot graph filter as a separate function. This allows to express pivot topic queries as composition (conjunction) of filters. Listing 6.4 explains the implementation by giving a piece of Python code of the *Pivot* class of the prototype exploration interface (Section 6.2). A *Pivot* object contains some important variables (line 3 to line 6), such as the variable $conf$ containing the workflow configuration, the variable $sparkConf$ containing the Spark configuration, the variable $future$ containing the future pivot table *Future* and the variable *past* containing the past pivot table *Past*. A *Pivot* object by default contains all future pivot topics and all past pivot topics. The function $emerge(kw)$ (line 11 to line 21) returns a new *Pivot* object which creates a new *Future* table and a new *Past* table containing future and past pivot topics with certain keywords in a given list *kw* as emerging terms. The function $array\_contains(col, value)$[1] on line 16 and line 17 is an SQL Array function of PySpark[2], which is used to check if an element value is present in an array

[1] https://spark.apache.org/docs/latest/api/python//reference/api/pyspark.sql.functions.array_contains.html

[2] https://spark.apache.org/docs/latest/api/python/index.html

type column on DataFrame. In the *emerge*(kw) function, we use it to find all pivot topics containing the emerging terms which exist in the term list kw. For other pivot topic filters, we adopt the same idea that each filter will return a new *Pivot* object.

Listing 6.4: Pivot class

```
class Pivot:
    def __init__(self, config, sparkConf, future = None, past = None):
        self.conf = config  # workflow configuration
        self.sparkConf = sparkConf  # Spark configuration
        self.future = future  # future pivot topic table
        self.past = past  # past pivot topic table
        ...
    ...

    # return pivot topics containing terms in kw as emerging terms
    def emerge(self, kw):
        listFuture = []
        listPast = []
        # for every term k in kw, find pivot topics containing k as emerging term
        for k in kw:
            listFuture.append(self.future.where(array_contains(self.future['emerging'], k)))
            listPast.append(self.past.where(array_contains(self.past['emerging'], k)))
        # union all pivot topic tables
        pivotFuture = reduce(lambda a,b : a.union(b), listFuture)
        pivotPast = reduce(lambda a,b : a.union(b), listPast)
        return Pivot(self.conf, self.sparkConf, pivotFuture, pivotPast)
    ...
```

6.2 EPIQUE Prototype

A first prototype [105] of the relational implementation of a generic topic evolution model is currently used to extract and analyze evolution patterns for different scientific domains in collaboration with philosophers of science. This prototype is implemented on top of Apache Spark for processing large scientific corpora containing millions of documents and finding meaningful topic evolution graphs for both stable topics and highly evolving ones. Section 6.2.1 introduces the architecture of the EPIQUE web application. Then Section 6.2.2 and Section 6.2.3 present two notebooks[3] for the generation and the exploration of pivot

[3] https://jupyter.org/

graphs through an example. A video demonstration of these two notebooks can be found on the following web site: `http://www-bd.lip6.fr/wiki/site/recherche/projets/epique/demo/start`.

6.2.1 Prototype Architecture

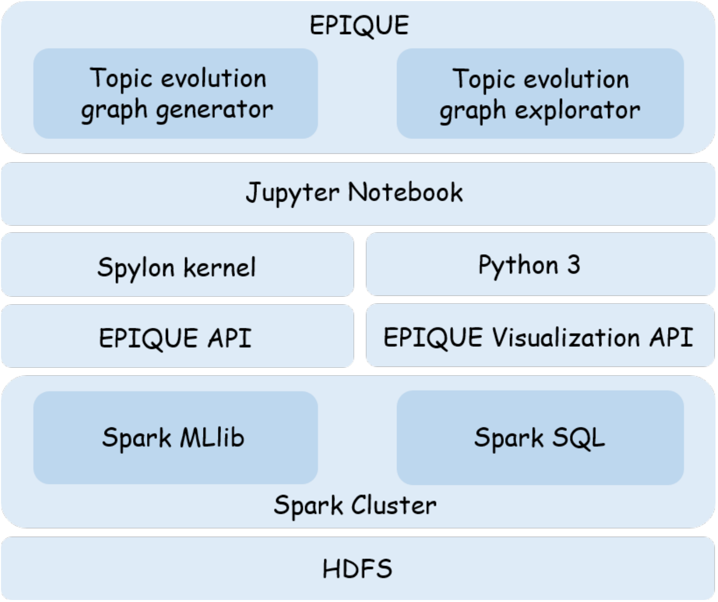


Figure 6.1: Architecture overview of EPIQUE web application

Figure 6.1 gives an overview of the architecture of our web application implemented on top of Apache Spark and Jupyter Notebook [106]. The application is accessible through two interactive notebooks for building and exploring the pivot evolution graphs.

The evolution graph generation application is implemented in Scala and the EPIQUE API provides all functions to execute the entire EPIQUE workflow as mentioned in Section 3.1. The *Topic evolution graph generator* notebook is executed by the Spylon[4] kernel which allows to use Python and Scala for interacting with Apache Spark through a notebook.

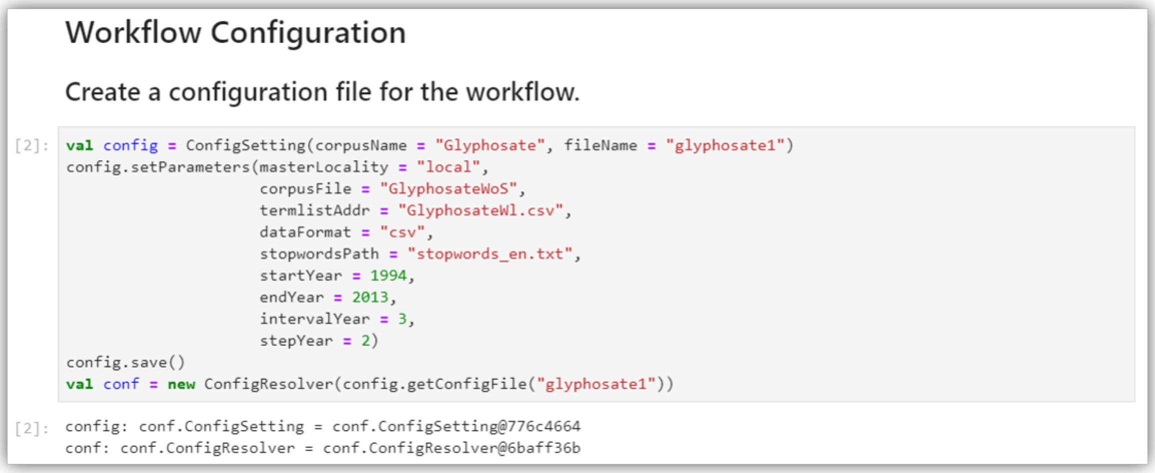
The *Topic evolution graph explorator* uses a standard Python kernel to take advantage of advanced Python-3 graphical user interface libraries (*ipywidgets*[5]) for facilitating user interaction with the EPIQUE Visualization API dedicated to the analysis and visualization of topic

[4] https://github.com/Valassis-Digital-Media/spylon-kernel

[5] https://ipywidgets.readthedocs.io/en/latest/

evolution graphs. The data storage is based on the Hadoop Distributed File System (HDFS[6]) which provides a software framework for distributed storage and processing of big data using the MapReduce programming model.

6.2.2 The *Topic Evolution Graph Generator* Notebook



Workflow Configuration

Create a configuration file for the workflow.

```
[2]: val config = ConfigSetting(corpusName = "Glyphosate", fileName = "glyphosate1")
     config.setParameters(masterLocality = "local",
                          corpusFile = "GlyphosateWoS",
                          termlistAddr = "GlyphosateWl.csv",
                          dataFormat = "csv",
                          stopwordsPath = "stopwords_en.txt",
                          startYear = 1994,
                          endYear = 2013,
                          intervalYear = 3,
                          stepYear = 2)
     config.save()
     val conf = new ConfigResolver(config.getConfigFile("glyphosate1"))
```

```
[2]: config: conf.ConfigSetting = conf.ConfigSetting@776c4664
     conf: conf.ConfigResolver = conf.ConfigResolver@6baff36b
```

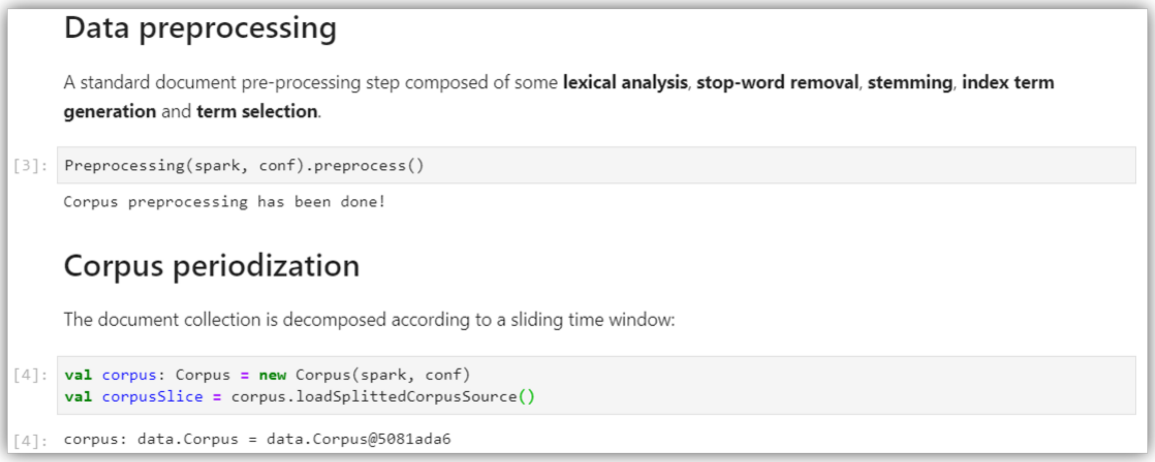
Figure 6.2: Screenshot: workflow configuration

To start the generation workflow, the user selects a corpus file (corpusFile), an optional vocabulary of terms (termlistAddr) pre-processed by an on-line text-mining tool Gargantext[7], a list of stopwords (stopwordsPath), the time period (startYear and endYear) and a sliding window (intervalYear and stopYear). Figure 6.2 gives the screenshot of the workflow configuration for processing the corpus "GlyphosateWoS" ranging from 1994 to 2013. The topics are computed for periods of 3 years, where each period overlaps with the preceding and the following period by two years.

Then the data preprocessing step and the corpus periodization step are executed as displayed in Figure 6.3 by applying the previous configuration settings.

As shown in Figure 6.4, the notebook allows users to visualize the number of documents per period. For example, the Glyphosate corpus is split into 10 periods and the number of documents in each period stays in the same order of magnitude.

[6] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[7] https://gargantext.org/



Data preprocessing

A standard document pre-processing step composed of some **lexical analysis**, **stop-word removal**, **stemming**, **index term generation** and **term selection**.

```
[3]: Preprocessing(spark, conf).preprocess()
```

```
Corpus preprocessing has been done!
```

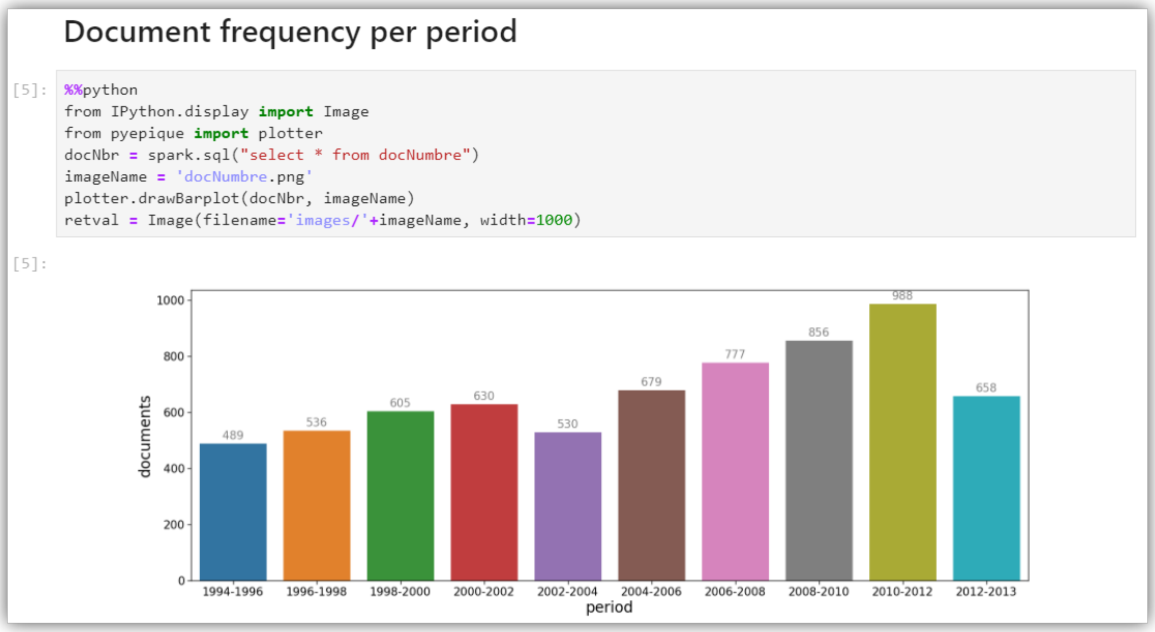
Corpus periodization

The document collection is decomposed according to a sliding time window:

```
[4]: val corpus: Corpus = new Corpus(spark, conf)
     val corpusSlice = corpus.loadSplittedCorpusSource()
```

```
[4]: corpus: data.Corpus = data.Corpus@5081ada6
```

Figure 6.3: Screenshot: data preprocessing and corpus periodization

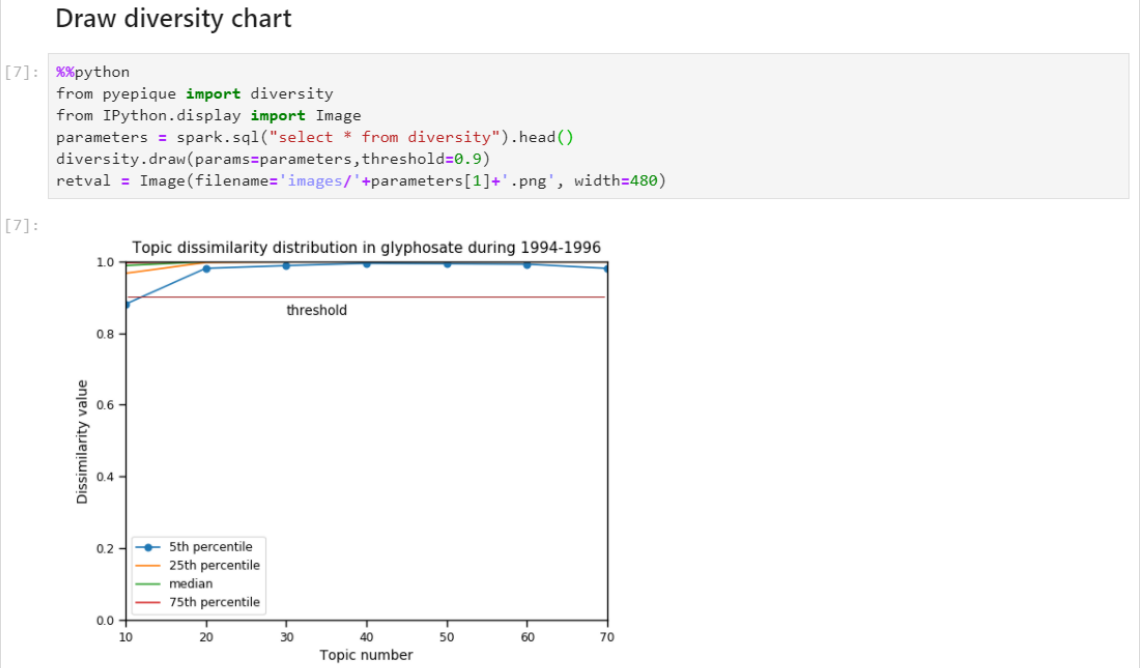


Document frequency per period

```
[5]: %%python
     from IPython.display import Image
     from pyepique import plotter
     docNbr = spark.sql("select * from docNumbre")
     imageName = 'docNumbre.png'
     plotter.drawBarplot(docNbr, imageName)
     retval = Image(filename='images/'+imageName, width=1000)
```

[5]:

Figure 6.4: Screenshot: amount of documents per period



Draw diversity chart

[7]:
```
%%python
from pyepique import diversity
from IPython.display import Image
parameters = spark.sql("select * from diversity").head()
diversity.draw(params=parameters,threshold=0.9)
retval = Image(filename='images/'+parameters[1]+'.png', width=480)
```

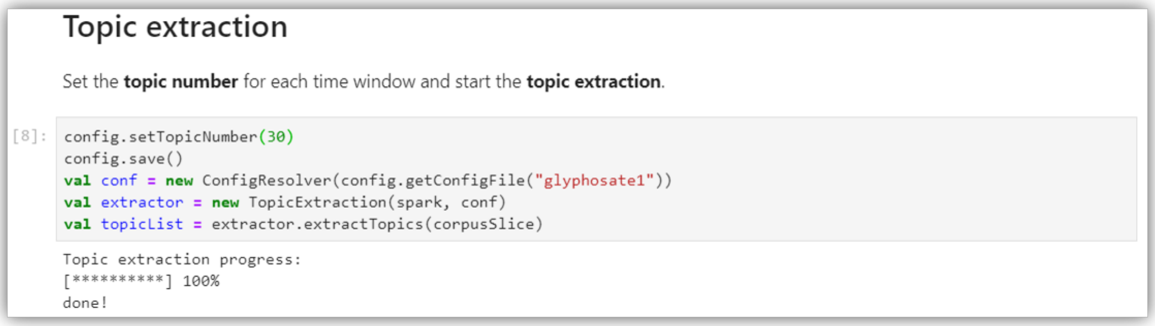
[7]:

Figure 6.5: Screenshot: topic diversity visualization

The LDA topic extraction step requires a vocabulary and a number of topics to be generated. This number obviously influences the diversity of the resulting topics. To choose a topic number for guaranteeing high topic diversity (Section 3.2.1), the application generates a set of topic models for different topic numbers. The user can then visualize the diversity (topic dissimilarity distribution) of the extracted topic models and choose the topic models with high-diversity for each period. A topic diversity distribution for different topic numbers is reported as shown in Figure 6.5 and, for example, by observing the 5^{th} percentile values (blue line), the user can retain one of the six models (20, 30, 40, 50, 60 or 70 topics per period) that achieves more than 95% of pairwise dissimilarities above 0.9.

After setting the LDA topic number for the workflow, the LDA topic model is executed sequentially *wrt*. the time windows and as displayed in Figure 6.6.

Finally, Figure 6.7 shows the alignment step where the extracted topics of consecutive periods are aligned by using cosine similarity. All pivot topic evolution graphs are generated along with their main temporal, structural and evolution indicators (*liveliness*, *relative evolution degree*, *pivot evolution degree*, *split degree*, *convergence degree*). All topic labels, such as *emerging*, *decaying*, etc. are also generated automatically in this step. The EPIQUE workflow is now



Topic extraction

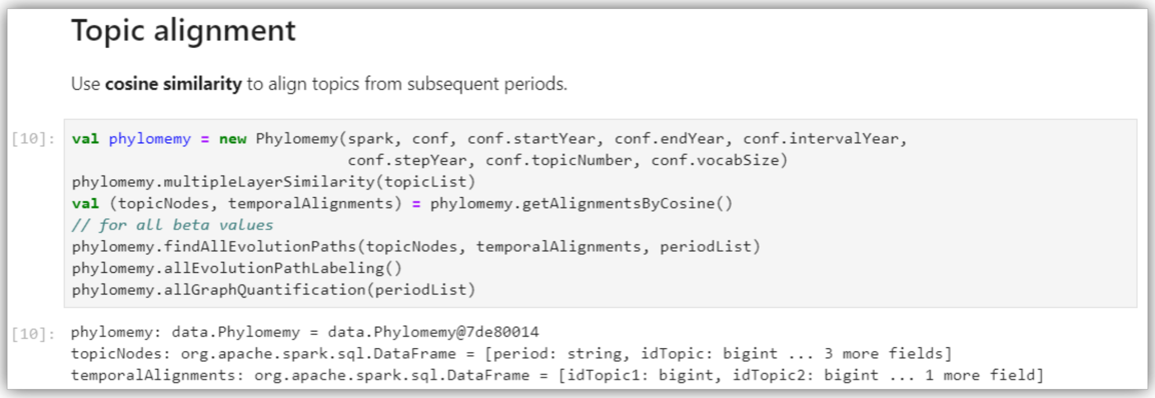
Set the **topic number** for each time window and start the **topic extraction**.

[8]:

```
config.setTopicNumber(30)
config.save()
val conf = new ConfigResolver(config.getConfigFile("glyphosate1"))
val extractor = new TopicExtraction(spark, conf)
val topicList = extractor.extractTopics(corpusSlice)
```

```
Topic extraction progress:
[**********] 100%
done!
```

Figure 6.6: Screenshot: topic extraction



Topic alignment

Use **cosine similarity** to align topics from subsequent periods.

[10]:

```
val phylomemy = new Phylomemy(spark, conf, conf.startYear, conf.endYear, conf.intervalYear,
                              conf.stepYear, conf.topicNumber, conf.vocabSize)
phylomemy.multipleLayerSimilarity(topicList)
val (topicNodes, temporalAlignments) = phylomemy.getAlignmentsByCosine()
// for all beta values
phylomemy.findAllEvolutionPaths(topicNodes, temporalAlignments, periodList)
phylomemy.allEvolutionPathLabeling()
phylomemy.allGraphQuantification(periodList)
```

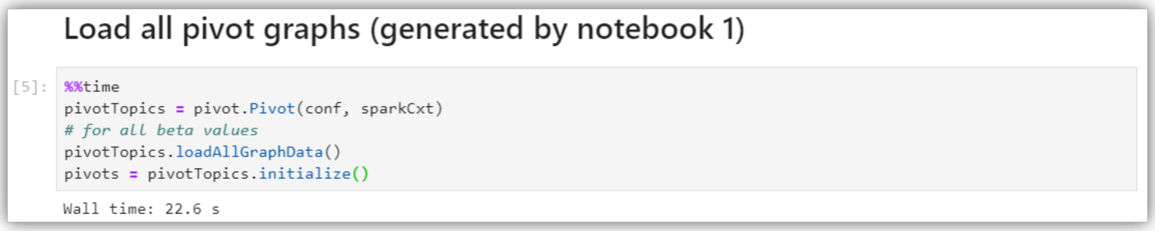
[10]:

```
phylomemy: data.Phylomemy = data.Phylomemy@7de80014
topicNodes: org.apache.spark.sql.DataFrame = [period: string, idTopic: bigint ... 3 more fields]
temporalAlignments: org.apache.spark.sql.DataFrame = [idTopic1: bigint, idTopic2: bigint ... 1 more field]
```

Figure 6.7: Screenshot: topic alignment

ready for exploration.

6.2.3 The *Topic Evolution Graph Explorator* Notebook



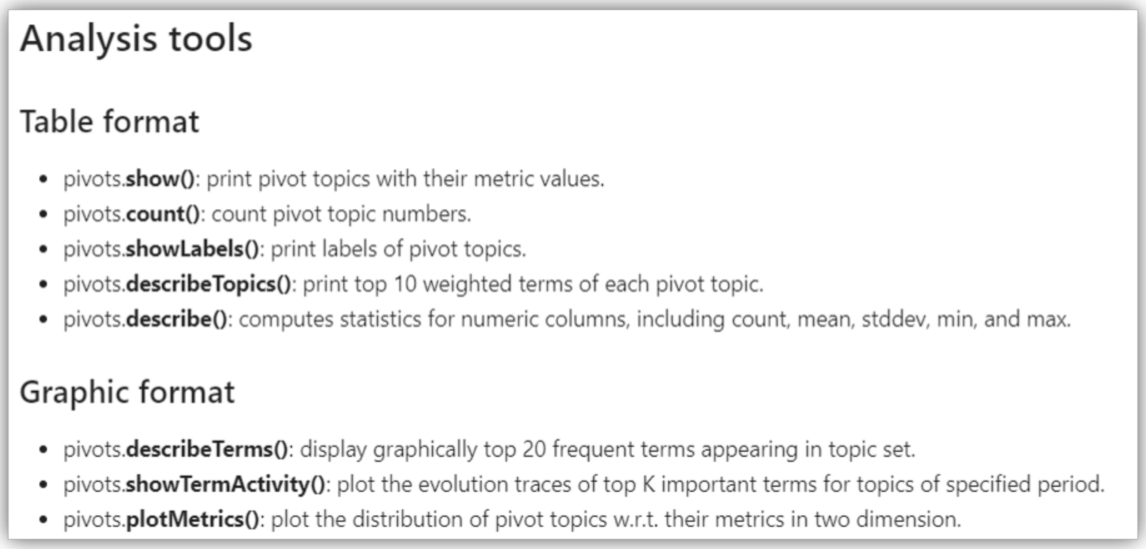
Load all pivot graphs (generated by notebook 1)

```
[5]: %%time
     pivotTopics = pivot.Pivot(conf, sparkCxt)
     # for all beta values
     pivotTopics.loadAllGraphData()
     pivots = pivotTopics.initialize()

     Wall time: 22.6 s
```

Figure 6.8: Screenshot: load all pivot graphs into memory

To start the pivot graph exploration process on the second notebook, the user loads all pivot graphs generated by the first notebook as shown in Figure 6.8. The final Dataframe *pivots* encapsulates all pivot graphs to be explored.



Analysis tools

Table format

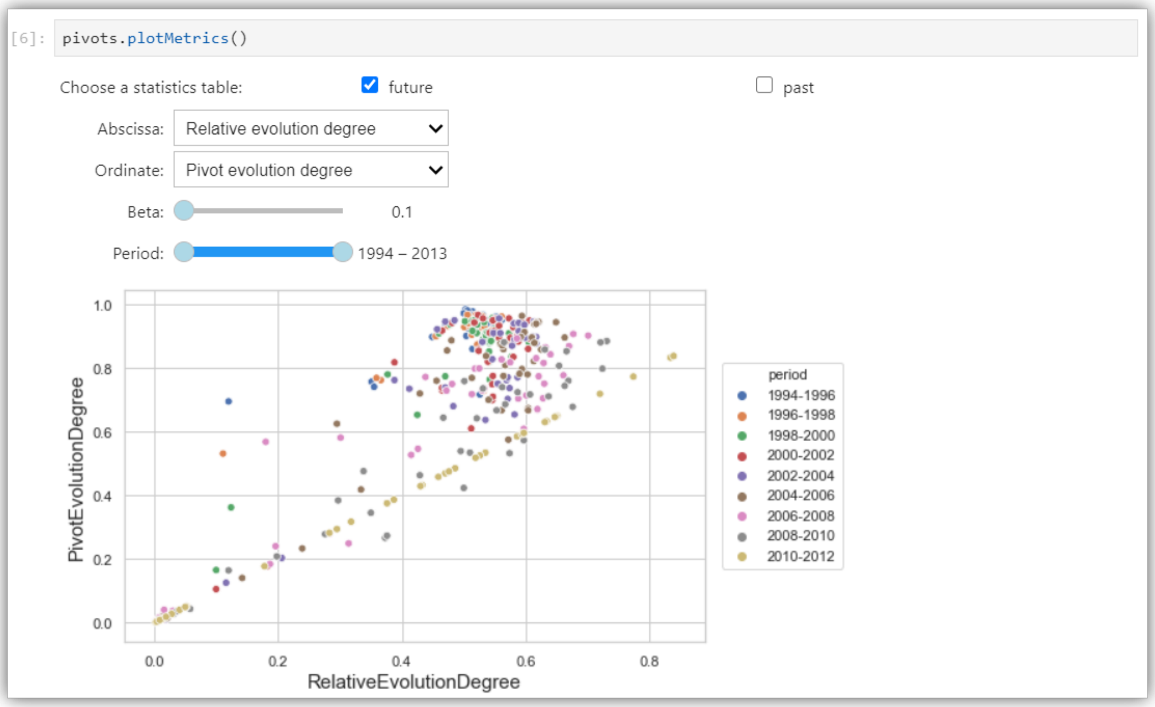
- pivots.**show()**: print pivot topics with their metric values.
- pivots.**count()**: count pivot topic numbers.
- pivots.**showLabels()**: print labels of pivot topics.
- pivots.**describeTopics()**: print top 10 weighted terms of each pivot topic.
- pivots.**describe()**: computes statistics for numeric columns, including count, mean, stddev, min, and max.

Graphic format

- pivots.**describeTerms()**: display graphically top 20 frequent terms appearing in topic set.
- pivots.**showTermActivity()**: plot the evolution traces of top K important terms for topics of specified period.
- pivots.**plotMetrics()**: plot the distribution of pivot topics w.r.t. their metrics in two dimension.

Figure 6.9: Screenshot: analysis tools

The notebook proposes some functions to analyse the content, the labels, the numbers and the term importance of pivot topics (fig. 6.9). Figure 6.10 is a graphical visualization of the pivot evolution degree and the relative evolution degree of all topics. The user can choose among all metrics defined in Figure 4.15.



```
[6]: pivots.plotMetrics()
```

Choose a statistics table: ☑ future ☐ past

Abscissa: Relative evolution degree

Ordinate: Pivot evolution degree

Beta: 0.1

Period: 1994 – 2013

Figure 6.10: Screenshot: an example of the analysis tools

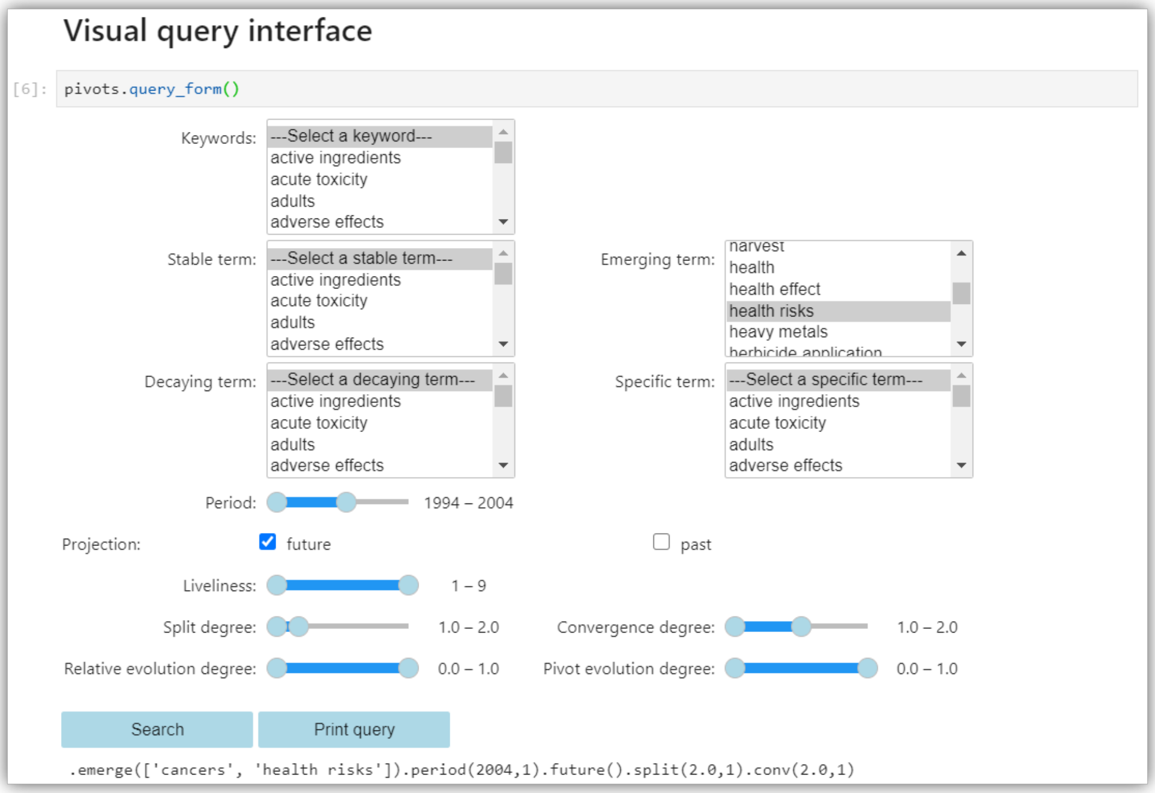
Figure 6.11 shows a graphical query interface where users can specify their exploration goal and visualize pivot topic evolution graphs.

In the next step, the user specifies the exploration goal through the query-by-example interface (as shown in the demonstration video[8]) and visualizes pivot topic evolution graphs as shown in Figure 6.12. The user can zoom out to study the structure of the evolution or zoom in to study the semantic changes.

The user also can express her exploration goal by combining different filters as defined in Section 4.5. Figure 6.13 shows the same results as displayed in Figure 6.12. The difference is that Figure 6.12 is obtained by executing a pivot topic query given through the user interface, whereas Figure 6.13 is obtained by directly running the same pivot topic query through the notebook.

Figure 6.14 gives another example of pivot topic query executed through the notebook. In this example, the user applies a *path* filter to select pivot topics (as shown in Figure 6.12) whose future contain topics having "children" as the stable label.

[8] http://www-bd.lip6.fr/wiki/site/recherche/projets/epique/demo/start



Visual query interface

```
[6]: pivots.query_form()
```

Keywords: ---Select a keyword--- / active ingredients / acute toxicity / adults / adverse effects

Stable term: ---Select a stable term--- / active ingredients / acute toxicity / adults / adverse effects

Emerging term: harvest / health / health effect / health risks / heavy metals / herbicide application

Decaying term: ---Select a decaying term--- / active ingredients / acute toxicity / adults / adverse effects

Specific term: ---Select a specific term--- / active ingredients / acute toxicity / adults / adverse effects

Period: 1994 – 2004

Projection: ☑ future ☐ past

Liveliness: 1 – 9

Split degree: 1.0 – 2.0

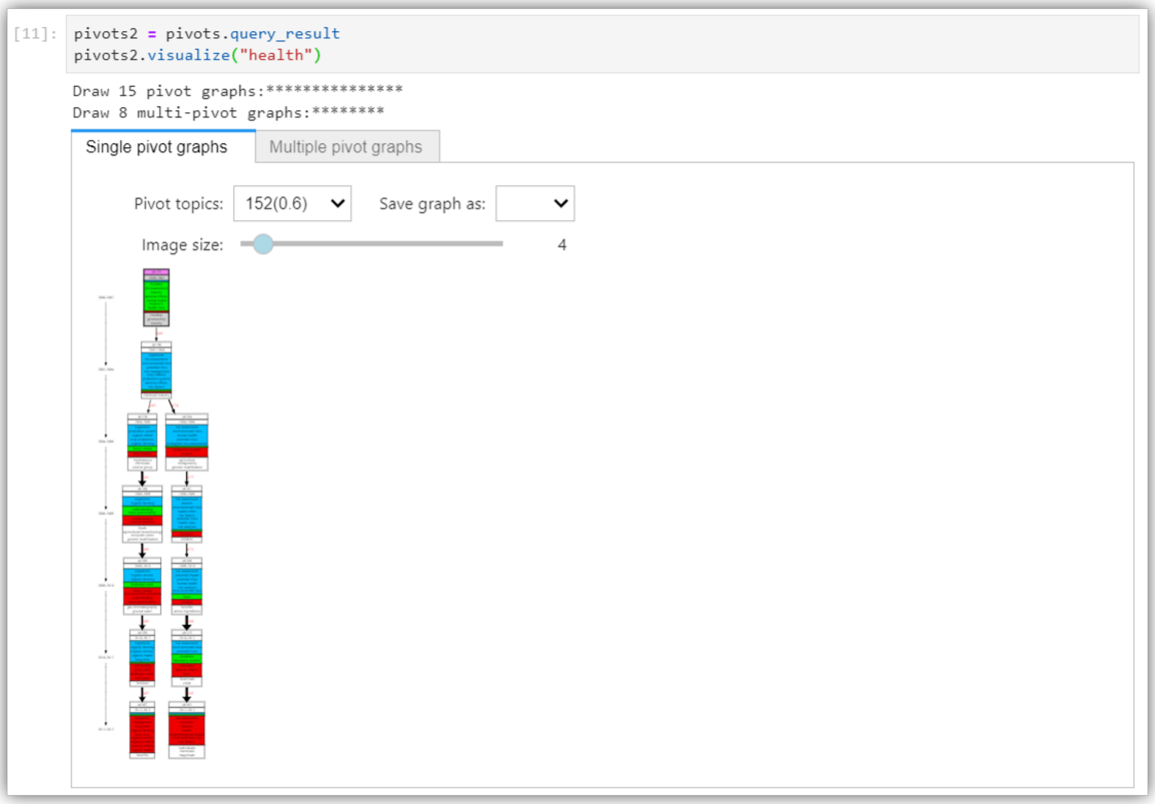
Convergence degree: 1.0 – 2.0

Relative evolution degree: 0.0 – 1.0

Pivot evolution degree: 0.0 – 1.0

```
.emerge(['cancers', 'health risks']).period(2004,1).future().split(2.0,1).conv(2.0,1)
```

Figure 6.11: Screenshot: query language user interface



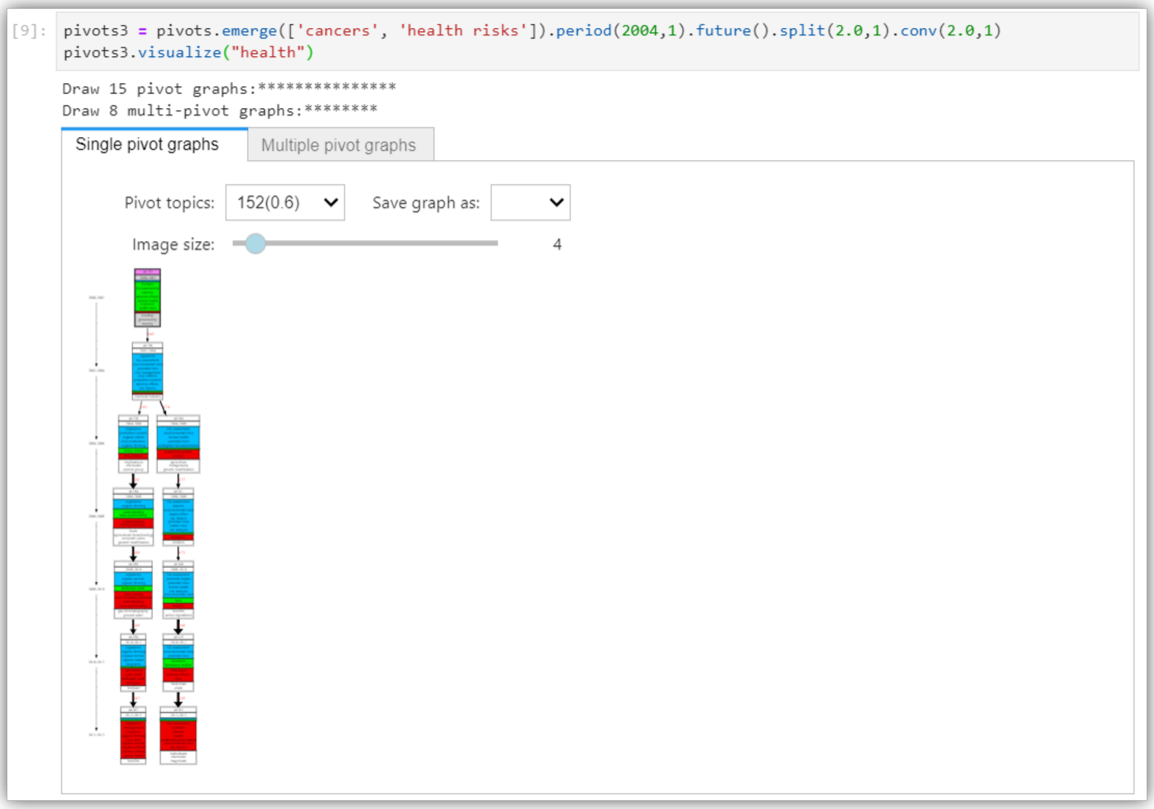
```
[11]: pivots2 = pivots.query_result
      pivots2.visualize("health")
```

```
Draw 15 pivot graphs:***************
Draw 8 multi-pivot graphs:********
```

Pivot topics: 152(0.6)

Image size: 4

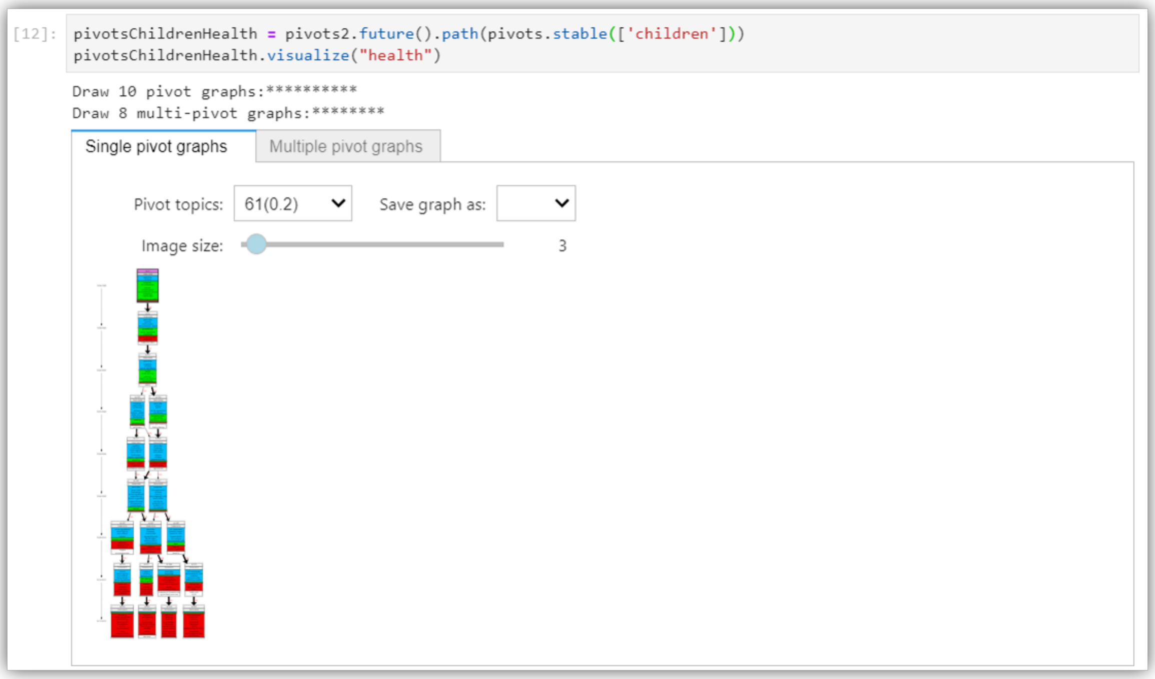
Figure 6.12: Screenshot: pivot topic evolution graph visualization



```
[9]: pivots3 = pivots.emerge(['cancers', 'health risks']).period(2004,1).future().split(2.0,1).conv(2.0,1)
     pivots3.visualize("health")

     Draw 15 pivot graphs:***************
     Draw 8 multi-pivot graphs:********
```

Figure 6.13: Screenshot: query pivot topic evolution graphs by filters



```
[12]: pivotsChildrenHealth = pivots2.future().path(pivots.stable(['children']))
      pivotsChildrenHealth.visualize("health")

      Draw 10 pivot graphs:**********
      Draw 8 multi-pivot graphs:********
```

Figure 6.14: Screenshot: query pivot topic evolution graphs by path filter

6.2.4 Some Pivot Graph Examples

Example 22 *Figure 6.15 presents the pivot graph as in Figure 6.12 in more detail. This pivot graph is filtered by the following user query: "select all pivot topics with cancers and health risks as emerging terms which appear before 2004.", which is shown in the user interface (Figure 6.11).*

The pivot topic $(t152, 0.6)$ containing terms about human health risks such as cancers and genotoxicity splits into two topics in the future where the first topic $(t238, 0.6)$ talks about the organic farming and the second one $(t204, 0.6)$ talks about health risk assessment.

Example 23 *Figure 6.16 presents a pivot graph $\mathcal{G}^{future}(t14, 0.4)$ extracted from the ISTEX corpus (see Table 3.1) by the following query: "select pivot topics containing economy and environmental quality which appear before 1995 with low split degree and low pivot evolution degree". As can be seen, the pivot topic $(t14, 0.4)$ talks about the relationship between natural resources (renewable resources, resource extraction, resource depletion, etc.), environmental quality and economy. In the future, pivot topic $(t14, 0.4)$ evolves into 3 topics (t150, t210 and t250). More precisely, topic t150 states policies including environmental policy and trade policy, topic t210 discusses the emission reduction and topic t250 focuses on forestry (forest managers, forest products, forest resources).*

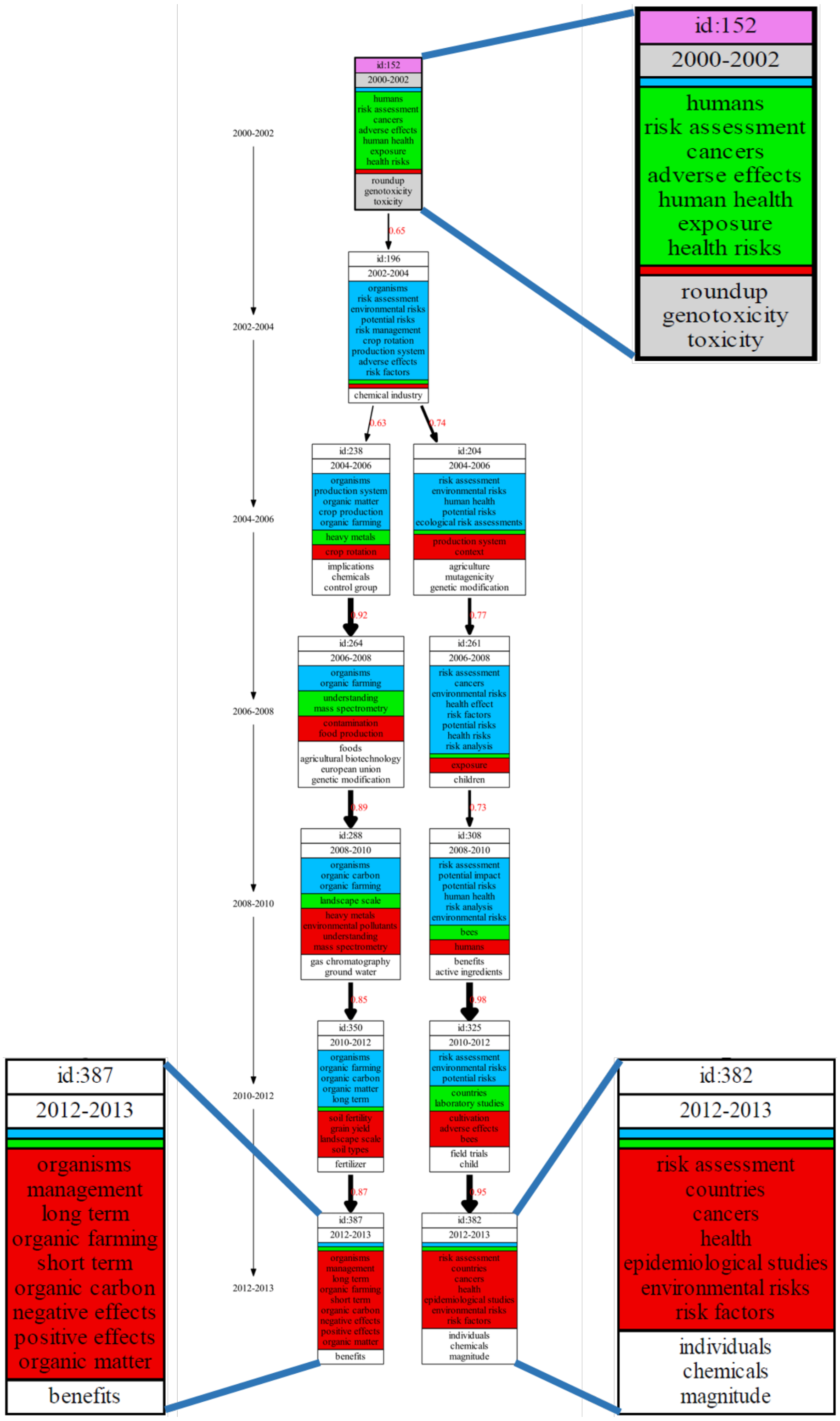


Figure 6.15: Pivot graph $\mathcal{G}^{future}(t152, 0.6)$ over the Glyphosate corpus

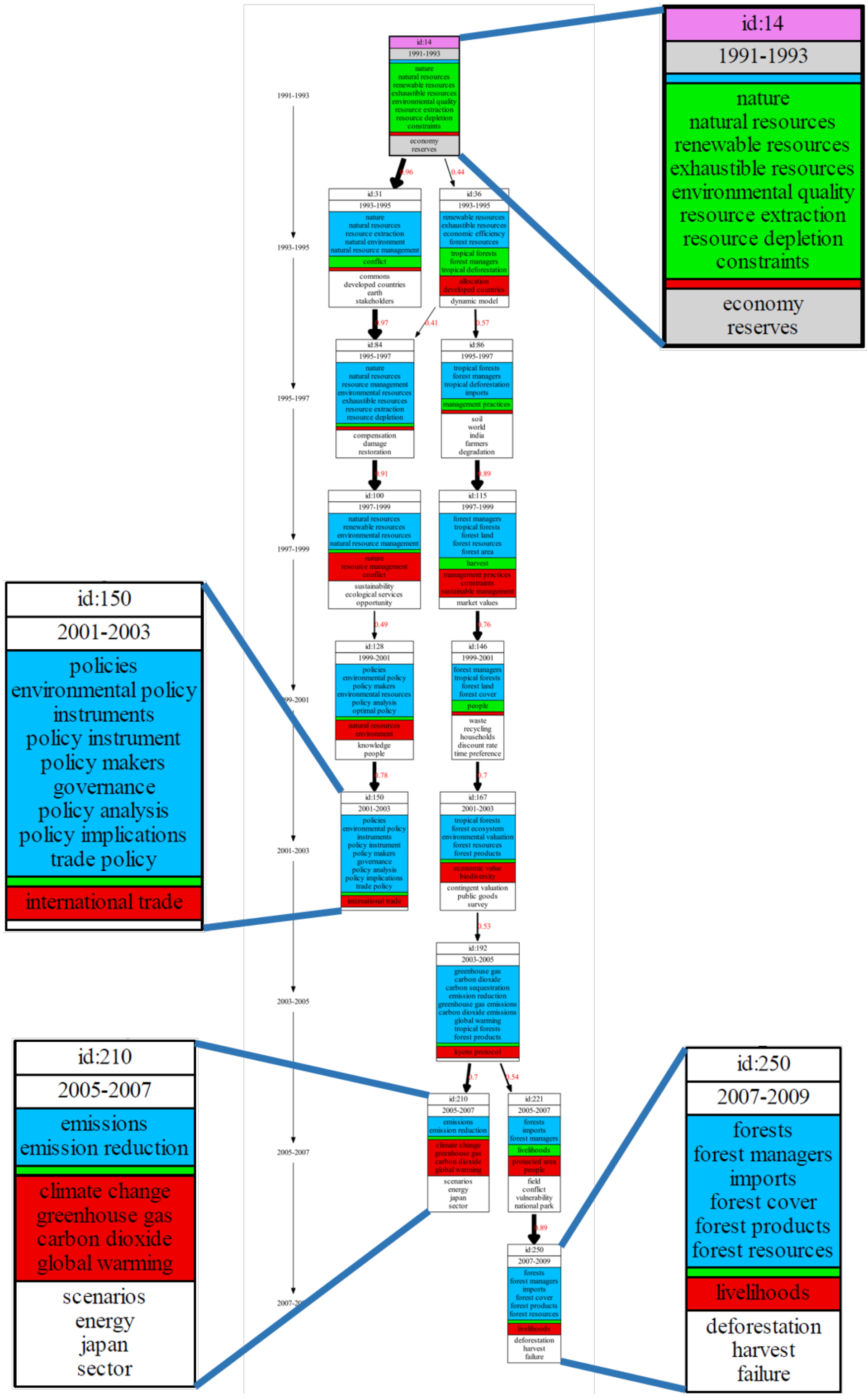


Figure 6.16: Pivot graph $\mathcal{G}^{future}(t14, 0.4)$ over the ISTEX corpus

Example 24 *The query which returns the pivot graph* $\mathcal{G}^*(t842, 0.2)$ *in Figure 6.17 filters all pivot topics containing images and features as the stable label with low relative evolution degree and low split and convergence degrees in the arXiv corpus. Pivot topic* $(t842, 0.2)$ *is evolved mainly from topic* $t299$ *which talks about data analysis. In the period from* 2012 *to* 2014*, topic* $t706$ *discussing "features" appears and merges with the future of topic* $t299$ *into pivot topic* $(t842, 0.2)$ *and then splits into two topics in the period from* 2016 *to* 2017 *where topic* $t942$ *is about feature engineering (accuracy, loss, gradient, etc.), whereas topic* $t978$ *talks about image processing (objects, images, matching, filtering, etc.).*

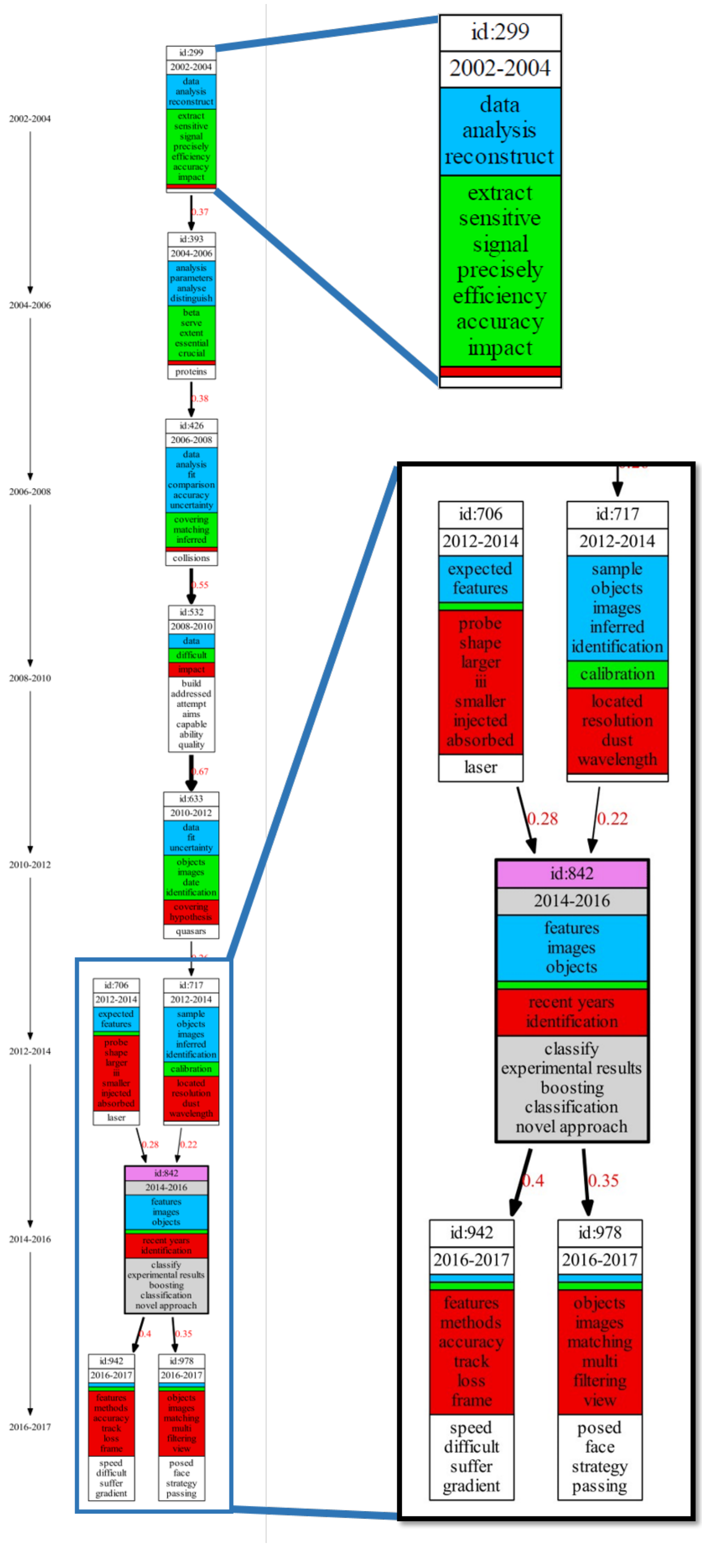


Figure 6.17: Pivot graph $\mathcal{G}^*(t842, 0.2)$ over the arXiv corpus

7 Conclusion and Future Work

In this thesis, we have presented the definition and implementation of a new framework for building and exploring topic evolution networks which represent the progress and evolution of research in scientific document archives. This framework is capable of extracting topic evolution networks representing the progress of science published in large scientific document archives. The obtained networks can be explored using a high-level query language based on the notion of pivot graphs which represent the evolution of topics at different levels of detail. The whole frameworks is implemented on top of Apache Spark to achieve scalability and can be accessed via two notebooks for the generation and the exploration of topic evolution networks.

7.1 Main Contributions

7.1.1 Topic Evolution Network Computation

The computation of topic evolution networks is mainly composed of two steps, a topic extraction step and a topic alignment step. For the topic extraction step, we apply the probabilistic topic model LDA [22], which describes each topic as a weighted term vector. LDA requires a fixed number of topics predefined by users. This parameter influences the diversity of the extracted topic set and the quality of the final phylomemy networks. The thesis addresses the challenge of the quality of topic sets by introducing a diversity-based measure (Section 3.2). The topic diversity within a period can be estimated by observing the dissimilarity distribution over all topic pairs within the period. Our experiments on the different scientific document archives have confirmed that the performance of LDA and the diversity of topics produced by LDA mainly depends on the size of the analyzed documents and this diversity-based measure can assist experts in choosing the optimal number of topics per period to produce highly divers phylomemies.

For the topic alignment step, we adopt the cosine similarity to align topics from different time periods. This alignment method performs well on measuring the correlations between sparse vectors. Moreover, our distributed map-reduce implementation of cosine similarity on top of Apache Spark has proved its efficiency for our high-dimensional weighted vector setting in the experimental evaluation.

7.1.2 Pivot Graph Model and Query Language

The data model of our framework relies on the notion of *pivot topic graphs* which represents the evolution of each individual topic by a set of multi-stage connected topic evolution subgraphs (Chapter 4). The pivot graph model includes a high-level filter-based query language (Section 4.5) which allows users to express complex evolution pattern queries composed of structural, temporal and semantic topic filters. We also implemented a notebook including a graphical query interface to filter and explore the evolution of topics in a simple and sound way.

7.1.3 Implementation and Optimization

A scalable proof-of-concept prototype [105] of our model has been implemented on top of Apache Spark and Jupyter Notebook using LDA for topic extraction, Spark SQL [104] for computing pivot topic graphs and PySpark for interactively exploring pivot topic graphs.

An important goal for the implementation of our framework was to achieve a highly-interactive exploration interface. To achieve low query processing time, we decided to apply a full materialization approach which consists in precomputing and storing all pivot graphs before starting the exploration phase. This precomputation includes the computation of topic alignment, and the materialization of all pivot graphs for different β thresholds including the pivot topic labels and the graph statistics (evolution degree, liveliness). The final result can be explored and visualized using a collection of filters that can directly be applied on the materialized graph properties.

The pivot graph generation is done off-line and on the distributed Apache Spark framework, and we've proposed several optimizations for the different precomputation steps.

Topic alignment computation: A topic evolution network might contain thousands of topics and alignment edges which are produced by comparing millions of topic pairs with some similarity function. We highlight how to adapt the native distributed cosine similarity computation implemented in Apache Spark to our particular context and to improve the performance by proposing two topic alignment strategies (Section 3.3). The first strategy (full matrix alignment) takes account of the similarities of all topic pairs which is important for our topic evolution model. The second alignment strategy (nearest-neighbor alignment) focuses on aligning topics only from consecutive periods and uses a nearest-neighbor condition to select candidate topics. We propose two distributed implementations which mainly differ on their ability to reduce data-shuffling during the alignment computation.

Pivot graph computation: Pivot graph computation mainly consists in extracting for each topic several connected subgraphs (pivot graphs) from the topic evolution network. The pivot graphs of a topic have different minimal edge weights (β thresholds) where any graph contains all other graphs with a higher minimal weight. We propose an efficient incremental join-based transitive closure algorithm for the materialization of pivot graphs and the graph properties in advance (Section 5.1.3). This algorithm benefits of the reduced number of SQL joins as well as the reduced size of joined data for the TC computation compared to the baseline approach (Section 5.1.1) and the smart approach (Section 5.1.2). To avoid many join operations of the join-based materialization strategy, we propose an alternative pivot graph materialization strategy based on GraphX [32] using Bulk Synchronous Parallel Paradigm [96].

7.2 Future Work

7.2.1 Generic Topic Evolution Workflow

A first direction for extending our current frameworks is to enrich the topic evolution model by exploiting the LDA document-topic matrix. Currently, we ignore these links between topics and documents which could be exploited for the topic-based exploration of document archives and for defining additional topic properties such as topic importance defined by the number of documents related to a given topic.

A second extension is to integrate other topic extraction and alignment methods as presented in Section 2.1 in order to generate and compare various types of topic evolution networks. The current implementation assumes the same number of topics in each time period. Our experiments show that we can in general obtain good results (in terms of diversity) for a given corpus where each period covers about the same number of documents. Other topic models which don't have this restriction (frequent itemset, clique percolation, HDP) are more flexible and might produce better results for more heterogeneous archives with highly varying document distributions in time.

The topics extracted by the current workflow are defined by a weighted set of terms. It is up to the expert to give a semantic interpretation of the topic according to his knowledge of the most relevant terms. This lack of semantic representation sometimes make it difficult to give a clear meaning to each topic. Thus, another objective of our future work is to extend our topic extraction and analysis task with knowledge graphs to allow for a more refined analysis of scientific domains and their evolution. The proposed approach might consist in combining text mining methods with semantic web technologies (RDF/SPARQL) [107] and knowledge graph resources like Wikidata[1], DBPedia[2] or Yago[3].

7.2.2 Topic Evolution Model and Query Language

Our current workflow is based on a complete materialization of all pivot graphs. This simplifies the query evaluation and achieves high query performance for interactive data exploration. However, this makes it more difficult to take account of possible updates of the topic evolution graph. For example, an expert might discover during the exploration of a graph that some topics are not relevant and should be modified or removed from the initial network. Such modifications currently imply a complete recomputation step. Another issue on the current implementation is the storage cost for storing all materialized pivot graphs. Therefore, we plan to study the implementation of pivot filters over the original topic evolution graphs (without a preliminary materialization step) which would avoid data preprocessing and data storage costs but also lead to higher query execution time. Nevertheless, we believe that there exist many optimization opportunities which can exploit the different algebraic properties (distributivity, commutativity, monotonicity) of pivot filters for optimizing complex conjunctive filter plans by cost-based query rewriting strategies.

To accelerate the pivot graph generation for the current materialization strategy by avoiding

[1] https://www.wikidata.org/wiki/Wikidata:Main_Page

[2] https://www.dbpedia.org/

[3] https://yago-knowledge.org/

many joins for the TC computation, the BSP-based pivot graph computation model (Section 5.2) will be integrated into the current prototype. Accordingly, since the BSP computation does not generate new pivot graphs but only changes the labels of the topic evolution graph nodes, this also needs a reimplementation of our pivot topic query language adapted to the BSP-based computation model. This implementation might exploit the existing property graph systems [108] like Neo4j [109] and the corresponding graph query languages Cypher [110] for expressive and efficient data querying in a property graph.

Bibliography

[1] H. Rostaing, N. Barts, and V. Lévsillé. "Bibliometrics: repressentation instrument of the multidisciplinary positioning of a scientific area. Implementation for and advisor scientific committee". In: *La interdisciplinariedad y la transdisciplinariedad en la organización del conocimiento cientıífico: actas del VIII Congreso ISKO-España. León, 18, 19 y 20 de abril de 2007*. Servicio de Publicaciones. 2007, pp. 341–352.

[2] B. R. Martin. "The evolution of science policy and innovation studies". In: *Research policy* 41.7 (2012), pp. 1219–1239.

[3] Y. Sun and V. Latora. "The evolution of knowledge within and across fields in modern physics". In: *Scientific reports* 10.1 (2020), pp. 1–9.

[4] K. Debackere and R. Veugelers. "The role of academic technology transfer organizations in improving industry science links". In: *Research policy* 34.3 (2005), pp. 321–342.

[5] D. B. Kell and E. Lurie-Luke. "The virtue of innovation: innovation through the lenses of biological evolution". In: *Journal of The Royal Society Interface* 12.103 (2015), p. 20141183.

[6] C. G. Figuerola, F. J. G. Marco, and M. Pinto. "Mapping the evolution of library and information science (1978–2014) using topic modeling on LISA". In: *Scientometrics* 112.3 (2017), pp. 1507–1535.

[7] L. Rocha, F. Mourão, H. Mota, T. Salles, M. A. Gonçalves, and W. Meira Jr. "Temporal contexts: Effective text classification in evolving document collections". In: *Information Systems* 38.3 (2013), pp. 388–409.

[8] P. Huneman. "Determinism, predictability and open-ended evolution: lessons from computational emergence". In: *Synthese* 185.2 (2012), pp. 195–214.

[9] J. Cat. "The unity of science". In: (2007).

[10] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. DiCuccio, R. Edgar, S. Federhen, et al. "Database resources of the national center for biotechnology information". In: *Nucleic acids research* 36.suppl_1 (2007), pp. D13–D21.

[11] T. S. Kuhn, O. Neurath, and T. S. Kuhn. *The Structure of scientific revolutions*. en. 2nd ed., enlarged. International encyclopedia of unified science Foundations of the unity of science ed.-in-chief: Otto Neurath ; Vol. 2 No. 2. OCLC: 258260085. Chicago, Ill: Chicago Univ. Press, 1994.

[12] H. Andersen, P. Barker, and X. Chen. *The cognitive structure of scientific revolutions*. Cambridge University Press, 2006.

[13] E. Garfield. "Citation Indexes for Science: A New Dimension in Documentation through Association of Ideas". en. In: *Science* 122.3159 (July 1955), pp. 108–111. ISSN: 0036-8075, 1095-9203. (Visited on 05/22/2019).

[14] X. Sun, J. Kaur, S. Milojević, A. Flammini, and F. Menczer. "Social Dynamics of Science". en. In: *Scientific Reports* 3 (Jan. 2013), p. 1069. ISSN: 2045-2322. (Visited on 05/22/2019).

[15] S. Fortunato, C. T. Bergstrom, K. Börner, J. A. Evans, D. Helbing, S. Milojević, A. M. Petersen, F. Radicchi, R. Sinatra, B. Uzzi, et al. "Science of science". In: *Science* 359.6379 (2018).

[16] Q. He, B. Chen, J. Pei, B. Qiu, P. Mitra, and L. Giles. "Detecting Topic Evolution in Scientific Literature: How Can Citations Help?" In: *ACM Conf. on Information and Knowledge Management*. CIKM '09. New York, NY, USA, 2009, pp. 957–966. ISBN: 978-1-60558-512-3. (Visited on 05/21/2019).

[17] D. Zhou, X. Ji, H. Zha, and C. L. Giles. "Topic Evolution and Social Interactions: How Authors Effect Research". In: *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*. CIKM '06. New York, NY, USA: ACM, 2006, pp. 248–257. ISBN: 978-1-59593-433-8. (Visited on 05/21/2019).

[18] L. Bolelli, S. Ertekin, D. Zhou, and C. L. Giles. "Finding topic trends in digital libraries". In: *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*. 2009, pp. 69–72.

[19] D. Chavalarias and J.-P. P. Cointet. "Phylomemetic patterns in science evolution—the rise and fall of scientific fields". In: *PloS one* 8.2 (2013), e54847. ISSN: 19326203.

[20] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. "Distributed representations of words and phrases and their compositionality". In: *arXiv preprint arXiv:1310.4546* (2013).

[21] H. Sayyadi and L. Raschid. "A graph analytical approach for topic detection". In: *ACM Transactions on Internet Technology (TOIT)* 13.2 (2013), pp. 1–23.

[22] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[23] P. Jaccard. "The Distribution of the Flora in the Alpine Zone.1". en. In: *New Phytologist* 11.2 (1912), pp. 37–50. ISSN: 1469-8137. (Visited on 11/28/2019).

[24] A. Singhal et al. "Modern information retrieval: A brief overview". In: *IEEE Data Eng. Bull.* 24.4 (2001), pp. 35–43.

[25] X. Sun, K. Ding, and Y. Lin. "Mapping the evolution of scientific fields based on cross-field authors". In: *Journal of Informetrics* 10.3 (Aug. 2016), pp. 750–761. ISSN: 1751-1577. (Visited on 05/21/2019).

[26] B. Hu, X. Dong, C. Zhang, T. D. Bowman, Y. Ding, S. Milojević, C. Ni, E. Yan, and V. Larivière. "A Lead-lag Analysis of the Topic Evolution Patterns for Preprints and Publications". In: *J. Assoc. Inf. Sci. Technol.* 66.12 (Dec. 2015), pp. 2643–2656. ISSN: 2330-1635. (Visited on 03/18/2019).

[27] T. White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[28] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, et al. "Spark: Cluster computing with working sets." In: *HotCloud* 10.10-10 (2010), p. 95.

[29] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. "Pregel: a system for large-scale graph processing". In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 2010, pp. 135–146.

[30] M. Bastian, S. Heymann, and M. Jacomy. "Gephi: an open source software for exploring and manipulating networks". In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 3. 1. 2009.

[31] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. "Graphviz—open source graph drawing tools". In: *International Symposium on Graph Drawing*. Springer. 2001, pp. 483–484.

[32] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica. “Graphx: A resilient distributed graph system on spark”. In: *First international workshop on graph data management experiences and systems*. 2013, pp. 1–6.

[33] D. Maurel, E. Morale, N. Thouvenin, P. Ringot, and A. Turri. “ISTEX: A database of twenty million scientific papers with a mining tool which uses named entities”. In: *Information* 10.5 (2019), p. 178.

[34] T. N. Rubin, A. Chambers, P. Smyth, and M. Steyvers. “Statistical topic models for multi-label document classification”. In: *Machine learning* 88.1-2 (2012), pp. 157–208.

[35] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. “Learning to classify short and sparse text & web with hidden topics from large-scale data collections”. In: *Proceedings of the 17th international conference on World Wide Web*. 2008, pp. 91–100.

[36] S. Hingmire, S. Chougule, G. K. Palshikar, and S. Chakraborti. “Document classification by topic labeling”. In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 2013, pp. 877–880.

[37] A. Perina, P. Lovato, V. Murino, and M. Bicego. “Biologically-aware latent dirichlet allocation (balda) for the classification of expression microarray”. In: *IAPR International Conference on Pattern Recognition in Bioinformatics*. Springer. 2010, pp. 230–241.

[38] B. Lu, M. Ott, C. Cardie, and B. K. Tsou. “Multi-aspect sentiment analysis with topic models”. In: *2011 IEEE 11th international conference on data mining workshops*. IEEE. 2011, pp. 81–88.

[39] A. C. Calheiros, S. Moro, and P. Rita. “Sentiment classification of consumer-generated online reviews using topic modeling”. In: *Journal of Hospitality Marketing & Management* 26.7 (2017), pp. 675–693.

[40] C. Wartena and R. Brussee. “Topic detection by clustering keywords”. In: *2008 19th International Workshop on Database and Expert Systems Applications*. IEEE. 2008, pp. 54–58.

[41] Z. Niu, G. Hua, L. Wang, and X. Gao. “Knowledge-based topic model for unsupervised object discovery and localization”. In: *IEEE Transactions on Image Processing* 27.1 (2017), pp. 50–63.

[42] D. Liu and T. Chen. “Unsupervised image categorization and object localization using topic models and correspondences between images”. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–7.

[43] J. Han, J. Pei, and Y. Yin. “Mining frequent patterns without candidate generation”. In: *ACM sigmod record* 29.2 (2000), pp. 1–12.

[44] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. “Uncovering the overlapping community structure of complex networks in nature and society”. In: *nature* 435.7043 (2005), pp. 814–818.

[45] D. M. Blei. “Probabilistic topic models”. In: *Communications of the ACM* 55.4 (2012), pp. 77–84.

[46] D. Chavalarias and J.-P. Cointet. “Bottom-up scientific field detection for dynamical and hierarchical science mapping, methodology and case study”. In: *Scientometrics* 75.1 (2008), pp. 37–50.

[47] A. A. Salatino, F. Osborne, and E. Motta. “AUGUR: Forecasting the Emergence of New Research Topics”. In: *ACM/IEEE on Joint Conference on Digital Libraries*. JCDL ’18. New York, NY, USA: ACM, 2018, pp. 303–312. ISBN: 978-1-4503-5178-2. (Visited on 03/25/2019).

[48] F. Osborne and E. Motta. “Mining semantic relations between research areas”. In: *International Semantic Web Conference*. Springer. 2012, pp. 410–426.

[49] I. Jeantet, Z. Miklós, and D. Gross-Amblard. “Overlapping Hierarchical Clustering (OHC)”. In: *International Symposium on Intelligent Data Analysis*. Springer. 2020, pp. 261–273.

[50] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. “Indexing by latent semantic analysis”. In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[51] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. “Singular value decomposition and principal component analysis”. In: *A practical approach to microarray data analysis*. Springer, 2003, pp. 91–109.

[52] B. Tang, M. Shepherd, E. Milios, and M. I. Heywood. “Comparing and combining dimension reduction techniques for efficient text clustering”. In: *Proceeding of SIAM international workshop on feature selection for data mining*. Citeseer. 2005, pp. 17–26.

[53] Y. Yang. “Noise reduction in a statistical approach to text categorization”. In: *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. 1995, pp. 256–263.

[54] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[55] T. Hofmann. "Probabilistic latent semantic indexing". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 1999, pp. 50–57.

[56] T. S. Ferguson. "A Bayesian analysis of some nonparametric problems". In: *The annals of statistics* (1973), pp. 209–230.

[57] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. "Sharing clusters among related groups: Hierarchical Dirichlet processes". In: *Advances in neural information processing systems*. 2005, pp. 1385–1392.

[58] J. W. Miller and M. T. Harrison. "A simple example of Dirichlet process mixture inconsistency for the number of components". In: *Advances in neural information processing systems*. 2013, pp. 199–206.

[59] A. Bhadury, J. Chen, J. Zhu, and S. Liu. "Scaling Up Dynamic Topic Models". In: *Proceedings of the 25th International Conference on World Wide Web*. WWW '16. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 381–390. isbn: 978-1-4503-4143-1. (Visited on 03/18/2019).

[60] D. M. Blei and J. D. Lafferty. "Dynamic Topic Models". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. New York, NY, USA: ACM, 2006, pp. 113–120. isbn: 978-1-59593-383-6. (Visited on 03/21/2017).

[61] C. Wang, D. Blei, and D. Heckerman. "Continuous Time Dynamic Topic Models". In: *Conference on Uncertainty in Artificial Intelligence*. UAI'08. Arlington, Virginia, United States: AUAI Press, 2008, pp. 579–586. isbn: 978-0-9749039-4-1. (Visited on 03/18/2019).

[62] X. Wang and A. McCallum. "Topics over Time: A non-Markov Continuous-time Model of Topical Trends". In: *Int'l Conf. on Knowledge Discovery and Data Mining*. KDD '06. New York, NY, USA: ACM, 2006, pp. 424–433. isbn: 978-1-59593-339-3. (Visited on 11/14/2018).

[63] H. Musoff and P. Zarchan. *Fundamentals of Kalman filtering: a practical approach*. American Institute of Aeronautics and Astronautics, 2009.

[64] J. Atchison and S. M. Shen. "Logistic-normal distributions: Some properties and uses". In: *Biometrika* 67.2 (1980), pp. 261–272.

[65] P. A. Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[66] S. Geman and D. Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[67] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: (1970).

[68] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker. "Perplexity—a measure of the difficulty of speech recognition tasks". In: *The Journal of the Acoustical Society of America* 62.S1 (1977), S63–S63.

[69] M. Franz, T. Ward, J. S. McCarley, and W.-J. Zhu. "Unsupervised and Supervised Clustering for Topic Tracking". In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '01. New York, NY, USA: ACM, 2001, pp. 310–317. ISBN: 978-1-58113-331-8. (Visited on 02/23/2017).

[70] S. Dharanipragada, M. Franz, J. S. McCarley, S. Roukos, and T. Ward. "Story segmentation and topic detection in the broadcast news domain". In: *Proceedings of the DARPA Broadcast News Workshop*. Citeseer. 1999, pp. 65–68.

[71] A. Kontostathis, L. M. Galitsky, W. M. Pottenger, S. Roy, and D. J. Phelps. "A survey of emerging trend detection in textual data mining". In: *Survey of text mining*. Springer, 2004, pp. 185–224.

[72] Y.-N. Tu and J.-L. Seng. "Indices of novelty for emerging topic detection". In: *Information processing & management* 48.2 (2012), pp. 303–325.

[73] T. L. Griffiths and M. Steyvers. "Finding scientific topics". en. In: *Proceedings of the National Academy of Sciences* 101.suppl 1 (Apr. 2004), pp. 5228–5235. ISSN: 0027-8424, 1091-6490. (Visited on 03/18/2019).

[74] H. Chipman, E. I. George, R. E. McCulloch, M. Clyde, D. P. Foster, and R. A. Stine. "The practical implementation of Bayesian model selection". In: *Lecture Notes-Monograph Series* (2001), pp. 65–134.

[75] D. Hall, D. Jurafsky, and C. D. Manning. "Studying the History of Ideas Using Topic Models". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 363–371. (Visited on 03/21/2017).

[76] B. Fuglede and F. Topsoe. "Jensen-Shannon divergence and Hilbert space embedding". In: *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* IEEE. 2004, p. 31.

[77] U. Cohen Priva and J. L. Austerweil. "Analyzing the history of Cognition using Topic Models". In: *Cognition*. The Changing Face of Cognition 135 (Feb. 2015), pp. 4–9. ISSN: 0010-0277. (Visited on 05/20/2019).

[78] Z. Zuo and K. Zhao. "A Graphical Model for Topical Impact over Time". In: *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*. JCDL '18. New York, NY, USA: ACM, 2018, pp. 405–406. ISBN: 978-1-4503-5178-2. (Visited on 03/18/2019).

[79] B. Chen, S. Tsutsui, Y. Ding, and F. Ma. "Understanding the topic evolution in a scientific domain: An exploratory study for the field of information retrieval". In: *Journal of Informetrics* 11.4 (2017), pp. 1175–1189. ISSN: 1751-1577. (Visited on 05/21/2019).

[80] A. Beykikhoshk, O. Arandjelović, D. Phung, and S. Venkatesh. "Discovering topic structures of a temporally evolving document corpus". In: *Knowledge and Information Systems* 55.3 (2018), pp. 599–632.

[81] A. Bhattacharyya. "On a measure of divergence between two statistical populations defined by their probability distributions". In: *Bulletin of the Calcutta Mathematical Society* 35 (1943), pp. 99–109. ISSN: 0008-0659. (Visited on 11/28/2019).

[82] V. Andrei and O. Arandjelović. "Complex temporal topic evolution modelling using the Kullback-Leibler divergence and the Bhattacharyya distance". In: *EURASIP Journal on Bioinformatics and Systems Biology* 2016.1 (2016), p. 16.

[83] E. Hellinger. "Neue Begründung der Theorie quadratischer Formen von unendlichvielen VerÃ¤nderlichen". German. In: *Journal für die Reine und Angewandte Mathematik* 136 (1909), pp. 210–271. ISSN: 0075-4102; 1435-5345/e. (Visited on 11/29/2019).

[84] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *Ann. Math. Statist.* 22.1 (Mar. 1951), pp. 79–86.

[85] F. Jian, W. Yajiao, and D. Yuanyuan. "Microblog topic evolution computing based on LDA algorithm". In: *Open Physics* 16.1 (2018), pp. 509–516.

[86] M. P. Deisenroth, A. A. Faisal, and C. S. Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[87] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: `http://www.aclweb.org/anthology/P/P14/P14-5010`.

[88] R. Grishman. *Computational linguistics: an introduction*. Cambridge University Press, 1986.

[89] M.-C. De Marneffe and C. D. Manning. *Stanford typed dependencies manual*. Tech. rep. Technical report, Stanford University, 2008.

[90] S. Petrov, D. Das, and R. McDonald. "A universal part-of-speech tagset". In: *arXiv preprint arXiv:1104.2086* (2011).

[91] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. "Mllib: Machine learning in apache spark". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1235–1241.

[92] J. Dean and S. Ghemawat. "MapReduce: Simplified data processing on large clusters". In: (2004).

[93] A. W. F. Edwards. *Likelihood*. CUP Archive, 1984.

[94] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-Graber, and D. M. Blei. "Reading tea leaves: How humans interpret topic models". In: *Advances in neural information processing systems*. 2009, pp. 288–296.

[95] R. V. Rasmussen and M. A. Trick. "Round robin scheduling–a survey". In: *European Journal of Operational Research* 188.3 (2008), pp. 617–636.

[96] L. G. Valiant. "A bridging model for parallel computation". In: *Communications of the ACM* 33.8 (1990), pp. 103–111.

[97] P. M. Kogge and H. S. Stone. "A parallel algorithm for the efficient solution of a general class of recurrence equations". In: *IEEE transactions on computers* 100.8 (1973), pp. 786–793.

[98] F. Bancilhon. "Naive evaluation of recursively defined relations". In: *On Knowledge Base Management Systems*. Springer, 1986, pp. 165–178.

[99] Y. E. Ioannidis. "On the computation of the transitive closure of relational operators". In: *VLDB*. Vol. 86. 1986, pp. 25–28.

[100] P. Valduriez and H. Boral. "Evaluation of Recursive Queries Using Join Indices". In: *Expert Database Conf.* 1986.

[101] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999.

[102] S. P. Borgatti. "Centrality and network flow". In: *Social networks* 27.1 (2005), pp. 55–71.

[103] D. Borthakur. "The hadoop distributed file system: Architecture and design". In: *Hadoop Project Website* 11.2007 (2007), p. 21.

[104] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, et al. "Spark sql: Relational data processing in spark". In: *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 2015, pp. 1383–1394.

[105] K. Li, H. Naacke, and B. Amann. "EPIQUE: Extracting Meaningful Science Evolution Patterns from Large Document Archives". In: *International Conference on Extending Database Technology (EDBT)*. demonstration. Copenhagen, Denmark, Mar. 2020, pp. 619–522.

[106] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.* Vol. 2016. 2016.

[107] J. Hebeler, M. Fisher, R. Blace, and A. Perez-Lopez. *Semantic web programming*. John Wiley & Sons, 2011.

[108] R. Angles. "The Property Graph Database Model." In: *AMW*. 2018.

[109] J. Webber. "A programmatic introduction to neo4j". In: *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*. 2012, pp. 217–218.

[110] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor. "Cypher: An evolving query language for property graphs". In: *Proceedings of the 2018 International Conference on Management of Data*. 2018, pp. 1433–1445.