An Analytic Graph Data Model and Query Language for Exploring the Evolution of Science

Ke Li, Hubert Naacke, Bernd Amann

Version of Record: https://www.sciencedirect.com/science/article/pii/S2214579621000642
Manuscript_efaad6bf93bbc8c422d956b8730072ce

An Analytic Graph Data Model and Query Language for Exploring the Evolution of Science

Ke Li[a,*], Hubert Naacke[a,*] and Bernd Amann[a,*]

[a]*LIP6, CNRS, Sorbonne Université, 4 Place Jussieu, 75005 Paris, France*

ARTICLE INFO

Keywords:
Topic Modeling
Topic Evolution Networks
LDA
Science Evolution
Big data

ABSTRACT

In this article we propose a data model and query language for the visualisation and exploration of topic evolution networks representing the research progress in scientific document archives. 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 first implementation of our model on top of Apache Spark and experimental results obtained for four real-world document archives.

1. Introduction

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)[1], arXiv[2], PubMed [39] or ISTEX[3]. 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 [33].

The evolution of scientific archives can broadly be studied by adopting a cognitive view or a social view on evolution dynamics. The *cognitive view* of scientific archive evolution emphasizes the shared knowledge and the change of ideas present in the document contents [22], whereas the *social view* takes account of authorship information and social interactions represented, for example, in co-authorship and citation graphs [13, 34]. There also exist methods which combine both views to study science evolution [16, 41]. In the interdisciplinary EPIQUE project[4], we assume that the evolution only depends on the textual document contents (title, abstract, main contents). The choice of the cognitive view reduces the number of analysis features, but it also decreases the "social" bias and detects more easily possible interactions between scientific ideas and contributions, independently of any particular scientific community.

Graph-based topic evolution analysis [20, 9, 1, 4] builds on topic evolution networks which track complex temporal evolution dynamics by applying topic discovery and topic alignment methods on a corpus of time-stamped documents. Figure 1 shows two snippets of a single topic evolution graph extracted from the arXiv[5] corpus using the EPIQUE system. The graph covers the years between 2000 and 2006 decomposed into three 3-year time periods overlapping by one year. The topic graph is obtained by applying the EPIQUE

*Corresponding author
ke.li@lip6.fr (K. Li); hubert.naacke@lip6.fr (H. Naacke); bernd.amann@lip6.fr (B. Amann)
ORCID(s): 0000-0003-3598-4129 (K. Li); 0000-0003-0559-9908 (H. Naacke); 0000-0002-6822-4049 (B. Amann)

[1]https://clarivate.com/webofsciencegroup/solutions/web-of-science/
[2]https://arxiv.org/
[3]https://www.istex.fr/
[4]This work was funded by French ANR-16-CE38-0002-01 project EPIQUE
[5]https://arxiv.org/

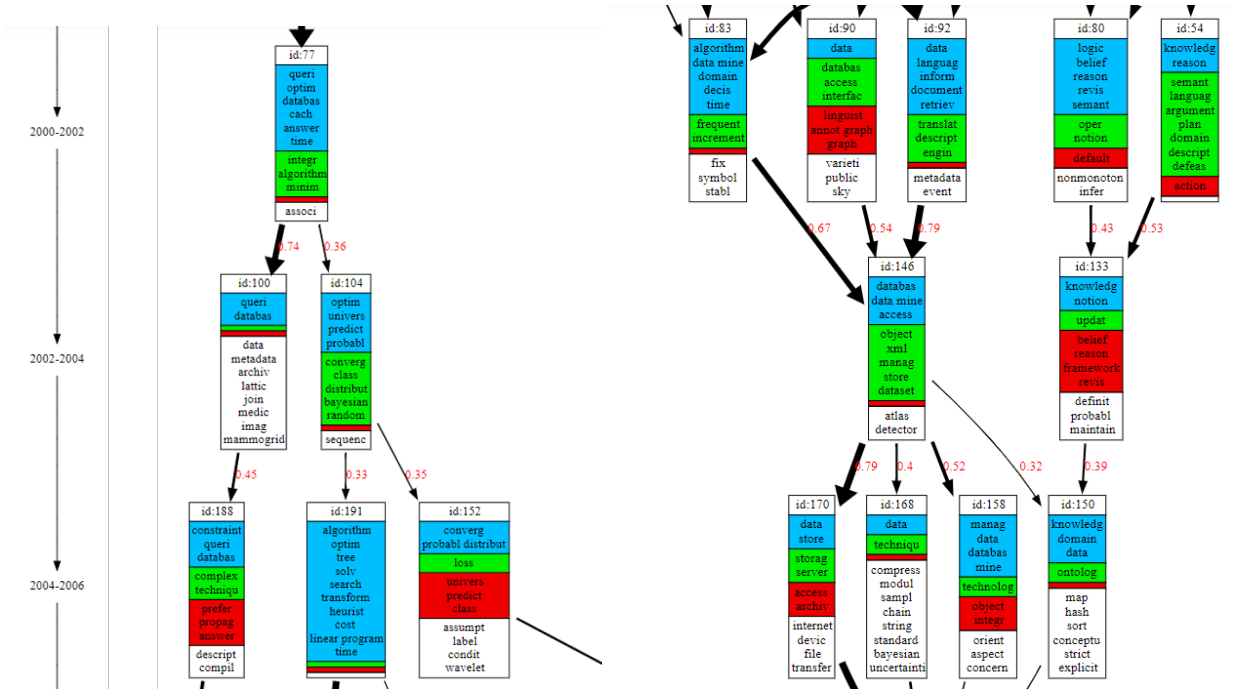


Figure 1: Pivot topics containing term "database" extracted from arXiv, green = emerging terms, blue = stable terms, red = decaying terms

workflow described in Section 5.1 to the subsets of documents covering each period. This workflow includes a standard NLP document pre-processing step (term extraction, stopword removal, stemming, term extraction) and applies the LDA (Latent Dirichlet Allocation) method for extracting a predefined number of weighted term vectors (topics) describing the scientific publication activity for each period. Finally, topics of two subsequent periods are connected by applying cosine similarity. Each topic is represented by a rectangle containing the top-10 weighted topic terms. *Emerging* terms are shown in green, *decaying* term boxes are colored in red, *stable* 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 cosine-similarity of the connected topic (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).

This article addresses two main issues when building and exploring topic evolution networks. First, building "meaningful" topic evolution networks is difficult and often an iterative process where domain experts must tune method-specific hyper-parameters and thresholds with respect to a given dataset and an expected output. This tuning process also includes the threshold-based filtering of topic alignment edges to reduce the network complexity. A second challenge concerns the visual exploration of large topic evolu-

tion networks. Whereas existing graph visualisation standards and tools like Gephi [6] or Graphviz [7] can be used to generate high-quality visualisations, their use for exploring large graphs and identifying meaningful evolution patterns is still limited.

In [25], 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 topic evolution networks. The graph model relies on the notion of *pivot topic graphs*, which describe the contents and the evolution dynamics of topics at different levels of detail. The model also 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. These topic filters include structural conditions on the evolution graph properties like average out- and in-degree, and temporal conditions on the topic term trends like term emergence and decay. To achieve a high level of interactivity, we choose to materialize these 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.

This article extends this work in different directions:

- A pivot topic calculus describing the formal semantics of the query language introduced in [25]. This extension also includes a formal analysis of the monotonicity properties of pivot filter expressions.
- An *optimized incremental transitive closure algorithm* for the materialization of pivot graphs. This algorithm is the core of the materialization process and implemented using Apache/Spark SQL.
- A detailed *experimental evaluation* of the performance of the different workflow steps including the LDA topic generation, the pivot graph computation, the topic labeling and the graph metrics computation over four real-world datasets. This evaluation also includes experiments measuring the scalability of our pivot graph generation algorithm over larger synthetic topic evolution graphs.

The remainder of this paper is organized as follows. The next section introduces the related work on topic evolution models and is followed by Section 3 which defines the EPIQUE pivot graph model. Section 4 describes our pivot graph query language including pivot topic functions and calculus. Section 5 gives an outline of the algorithms for building topic evolution networks and explains our query evaluation strategy. In Section 6 we detail the implementation of pivot graph computation. Then in Section 7, some experimental results obtained by applying our pivot graph model on four different document archives will be illustrated. The final section presents our conclusions and outlines future work.

2. Related Work

2.1. Topic Modeling

Topic modeling is an unsupervised text mining task which consists of extracting a compact representation of the contents represented in one or several documents. Topics are in general represented by groups of unweighted or weighted terms. Topic modeling helps in document classification [31], sentiment analysis [26], topic discovery [8], image object localization [30], etc.

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. In other words, the semantics of a

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

document is actually governed by some latent variables and the goal of topic modeling is to uncover these latent variables that shape the meaning of the document and the whole corpus.

Statistical topic models like probabilistic latent semantic analysis (pLSA) [17] use a probabilistic method to generate documents as mixtures of a low-dimensional set of topics. LSA models the probability $P(W, D)$ of each (word, document) co-occurrence as a mixture of conditionally independent multinomial distributions as shown in the following formula with Z being the set of estimated topics:

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

However, in this formula, 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. LDA [8] is a Bayesian version of pLSA procedure and follows the intuition that the probability distribution over words is skewed and apply a sparse Dirichlet prior to model the per-document topic and per-topic word distributions. LDA 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. Hierarchical Dirichlet Process (HDP) [35] 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, but the number of topics also becomes a random variable generated by a Dirichlet process. However, it has been shown that HDP is inconsistent for estimating the number of topics even with an infinite amount of data [28]. Furthermore, HDP does not scale to very large document corpus.

The goal of dynamic topic models [5, 7, 36, 38] 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. In our work, we are interested in generating and exploring topic graphs 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 Detection and 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. The topic detection and tracking system of [21] aims to identify and follow event-based topics across incoming streams of documents. Usually, a tracking system is given seed documents to monitor the document stream for further documents on the same topic, whereas a detection system performs an unsupervised clustering of the incoming document stream. [12] describes a tracking system which converts an unsupervised Topic Detection system into a supervised Topic Tracking system by sharing the confidence score. In a sense, it is the conversion of the output of a detection system into the output of a tracking system. A Gibbs Sampling based implementation of LDA has been applied by [14] 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 and studies the evolution of topics by applying a linear trend analysis on the mean θ values (document distributions over the topics) by year. Hu et. al. [18] applied LDA and regression analysis to identify different topic evolution patterns for preprints and papers from arXiv

and the Web of Science (WoS) in astrophysics for the last 20 years (1992 − 2011). The authors redefine the notion of topic trend and popularity, and demonstrate 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. Breakthrough research may not be the mainstream area, but can attract a significant amount of citations. For that reason, [42] proposes Topical Impact over Time (TIoT) which can be used for detecting trending topics and suggesting impactful papers in a bibliographical database. They applied collapsed Gibbs sampling for approximate inference and citation counts to quantify topical impact.

Our work models the evolution of topics by subgraphs connecting similar topics from different time periods without measuring and comparing their distribution, popularity and impact. More generally, trend analysis describes the temporal evolution of the popularity, the utility or the interest of topics, but does not take account of their structural evolution where one topic can evolve into several sub-topics or several topics can merge into a single topic.

2.3. Topic Evolution Networks

Topic evolution networks represent the structural topic evolution and track complex temporal changes by periodic topic discovery and directed acyclic networks aligning topics of different periods. Existing evolution network based frameworks mainly can be distinguished by the chosen topic extraction and alignment methods.

Instead of characterizing the evolving topics at fixed time points, [20] defines a topic as a quantized unit of evolutionary change in content and identifies topics along with the time that they start to appear in the corpus. They use the mixture of word distributions to discover topics and then connect topics to form a topic evolution graph using a measure derived from the underlying document network such as the citation network. [9] comes up with a method to enable a bottom-up reconstruction of the dynamics of scientific fields. They generate topics by word co-occurrence graphs and align inter-temporal topics by Jaccard similarity [19]. [1] generates topics by a Hierarchical Dirichlet Process (HDP) [35] and uses Bhattacharyya similarity [6], representing the gradual speciation and convergence similar to biologic evolution, for identifying topic alignments. The alignment process also applies (asymmetric) Kullback-Leibler divergence (KLD) [23] for detecting topic split and merge. [4] 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, BHD and Jaccard similarity. 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. [32] introduces a novel approach to the early detection of research topics by using the Computer Science Ontology[8] to model research topics in the Rexplore system. They apply a Clique Percolation Method (ACPM) for analyzing the dynamics between existent topics.

Other examples of science evolution studies apply unsupervised topic modeling to the ACL Anthology to analyze historical trends in the field of Computational Linguistics from 1978 to 2006 [15]. How "cognitive science" as a field has changed over the last three decades has been explored by [11]. And [10] analyzes topic evolution patterns (split, merge and knowledge transfer) in the field of Information retrieval (IR).

Our article extends previous works on topic evolution networks by a formal query language based on a set of semantic and structural graph filters to define complex topic evolution patterns.

3. Pivot Graph Model

This section 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.

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

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, $E = \{(t_i, t_j) \in T | sim(t_i, t_j) > 0 \wedge t_j.period = t_i.period + 1\}$.*

Example 1. *In Figure 1, P has 3 periods: $p1$="2000 − 2002", $p2$="2002 − 2004" $p3$="2004 − 2006", T_{p1} contains topics $t54$ to $t92$, and topic $t77 = (v, p1)$, where v is a weighted vector with positive weights for terms "queri", "optim", "databas", etc. Each topics is labeled by its top-10 highest ranked terms and topic similarity is estimated by the cosine similarity between the corresponding two term vectors. The similarity between topic $t77$ and topic $t100$ is $sim(t77, t100) = 0.74$.*

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 $|t.period - t'.period| > 1$. 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 4.1 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 over time.

Definition 3 (pivot topics and pivot evolution graphs). *Let $t \in T$ 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 $(t', t'') \in E'$.*

We distinguish three particular pivot evolution graphs among all possible pivot graphs of a given pivot topic (t, β), :

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).*

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.*

Example 2. *The topic spectrum of the left graph in Figure 1 contains values* 0.33, 0.35, 0.36, 0.45 *and* 0.74.

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. Pivot Graph Query Language

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.

4.1. Pivot Topic Functions

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:

$$period(t) = t.period \tag{1}$$

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 $t.terms$:

$$labels(t) = top_k(V)\text{: top-k terms in } V \text{ ranked by vector } t.terms \tag{2}$$

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 pivot evolution graph past $\mathcal{G}^{past}(t,\beta)$ 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 Function: The following function computes the set of topics that appear in the past and in the future of (t,β), respectively:

$$path^{\delta}(t,\beta) = \{t' | t' \text{ topic in } \mathcal{G}^{\delta}(t,\beta)\} \tag{3}$$

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):

$$labels^{\delta}(t,\beta) = \{l | t' \in path^{\delta}(t,\beta) \wedge l \in labels(t')\} \tag{4}$$

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:

$$emerge(t,\beta) = labels^{future}(t,\beta) - labels^{past}(t,\beta) \tag{5}$$

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

$$decay(t,\beta) = labels^{past}(t,\beta) - labels^{future}(t,\beta) \tag{6}$$

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

$$stable(t,\beta) = labels^{future}(t,\beta) \cap labels^{past}(t,\beta) \tag{7}$$

- Specific terms which neither exist in the past nor in the future topics of t:

$$specific(t,\beta) = t.labels - (labels^{future}(t,\beta) \cup labels^{past}(t,\beta)) \tag{8}$$

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

$$live^{\delta}(t,\beta) = max\{length(path) | path \text{is a path in } \mathcal{G}^{\delta}(t,\beta)\} \tag{9}$$

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.

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

$$revol^{\delta}(t,\beta) = 1 - avg_{(t_i,t_j)\in E}(sim(t_i,t_j)) \tag{10}$$

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.

$$pevol^{\delta}(t,\beta) = 1 - avg_{t_i\in T}(sim(t,t_i))) \tag{11}$$

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 .

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

$$split^{\delta}(t, \beta) = \frac{|E|}{|\{t_i | t_i \in T \wedge (t_i, t_j) \in E\}|} \tag{12}$$

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.

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

$$conv^{\delta}(t, \beta) = \frac{|E|}{|\{t_j | t_j \in T \wedge (t_i, t_j) \in E\}|} \tag{13}$$

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.

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').

Example 3. *For example, $live^{\delta}$ 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 $live^{\delta}(t, \beta') <= live^{\delta}(t, \beta)$. Similarly, 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 $\beta \leq \beta'$. This follows from 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 $\beta \leq \beta'$ and using the definition of specific. Functions $labels$ and $period$ are constant (return the same local label and period independently of β), $path^{\delta}$, $labels^{future}$ and $labels^{past}$ filters are monotonically decreasing.*

4.2. 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 $evol^{\delta}$ 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;*

Table 1
Monotonicity propagation 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

- *The expression period ϕ 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 $\neg$.*

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 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 4.1 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[9]. 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[10]. Finally, if F is constant or non-monotonic, the corresponding filters are static or non-monotonic.

Table 1 summarizes the monotonicity propagation for Boolean combinations of monotonic, anti-monotonic, non monotonic and static predicates.

Example 4. *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 $l \notin labels^{future} \wedge l \in labels^{past}$ is monotonic). The filter predicate stable*

[9]For all (t, β), if $l \in F$ holds for (t, β) then $l \in F$ holds for all (t, β') where $\beta' \geq \beta$.
[10]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 2
Pivot Filter Expressions

Expression F	Semantics $\overline{F}$	Monotonicity	Expression F	Semantics $\overline{F}$	Monotonicity ($\phi = \leq$)
Contains(L)	$\exists l \in L : l \in labels$	static	**Live** ϕ c	$live^{\delta}\ \phi\ c$	anti-monotonic
Emerge(L)	$\exists l \in L : l \in emerge$	non monotonic	**Revol** ϕ c	$revol^{\delta}\ \phi\ c$	non monotonic
Decay(L)	$\exists l \in L : l \in decay$	non monotonic	**Pevol** ϕ c	$pevol^{\delta}\ \phi\ c$	non monotonic
Stable(L)	$\exists l \in L : l \in stable$	anti-monotonic	**Split** ϕ c	$split^{\delta}\ \phi\ c$	non monotonic
Specific(L)	$\exists l \in L : l \in specific$	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 1
Minus(F_1)	$\neg\overline{F_1}$	see Table 1
F_1.**Union**(F_2)	$\overline{F_1} \vee \overline{F_2}$	see Table 1
Path(F_1)	$\exists t : t \vDash \overline{F_1} \wedge t \in path^{\delta}$	anti-monotonic

b Complex filter expressions

Expression	Semantics
Future	$\delta := future$
Past	$\delta := past$

c Graph projection

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 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 $\beta' < \beta$).

4.3. 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 4.2.

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 2a. 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 2b. Finally, the query language defines two projection operators **Future** and **Past** (Table 2c) 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.*

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

Example 5. *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: $DB.\textbf{Future}.\textbf{Revol}(>=0.5).\textbf{Pevol}(>=0.6).\textbf{Split}(>=2).\textbf{Live}(=5)$

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

Parameter δ is by default equal to future.

The result of query $Q1$ is shown in Figure 2.

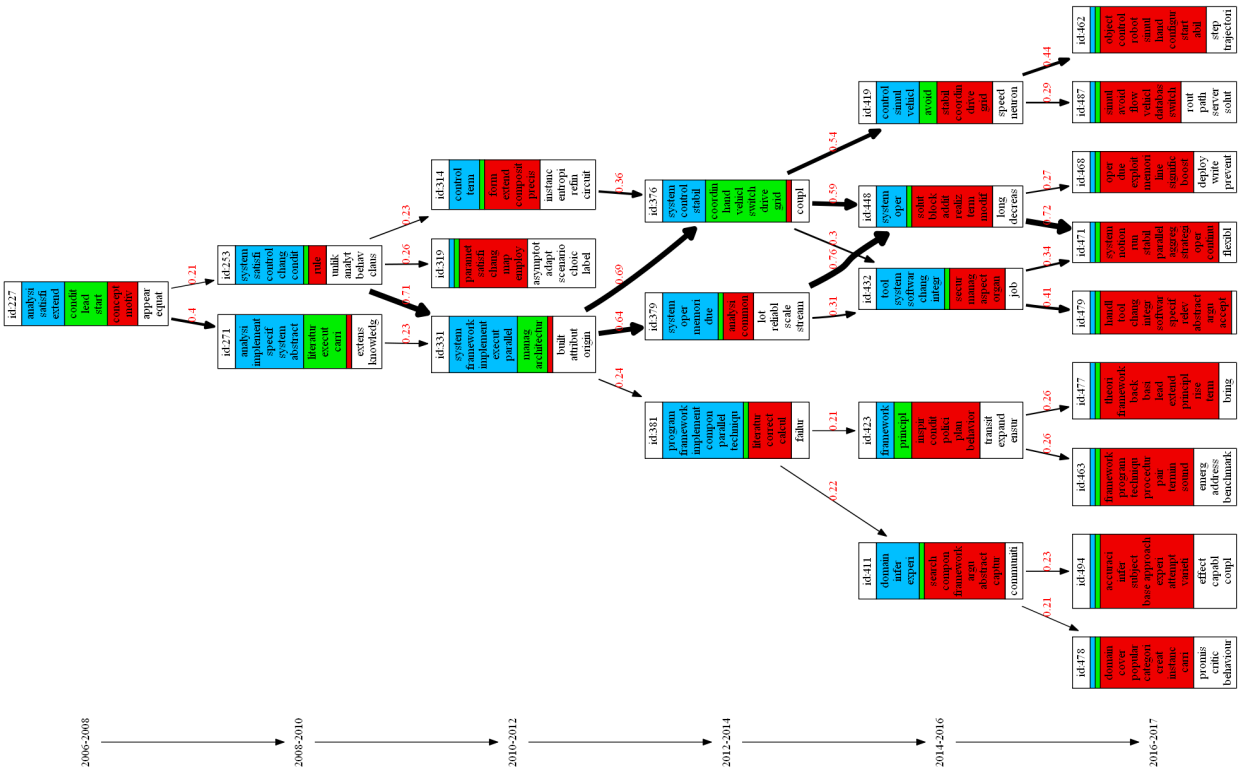


Figure 2: $\overline{Q1}: revol^{future} \geq 0.5 \wedge pevol^{future} \geq 0.6 \wedge split^{future} \geq 2 \wedge live^{future} = 5$

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 the decaying term "big data" and without emerging term "deep learning":*

  Q2: $DB.\textbf{Decay}(\texttt{"big\_data"}).\textbf{Minus}(\textbf{Emerge}(\texttt{"deep\_learning"}))$

  $$\overline{Q2}: \text{“big data”} \in decay \wedge \text{“deep learning”} \notin emerge$$

- *Find all topics with an emerging term "deep learning" where the past contains a path to a topic with the decaying term "big data":*

  Q3: $DB.\textbf{Emerge}(\texttt{"deep\_learning"}).\textbf{Past}.\textbf{Path}(\textbf{Decay}(\texttt{"big\_data"}))$

  $$\overline{Q3}: \text{“deep learning”} \in emerge.path^{past}(\text{“big data”} \in decay)$$

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

Three other queries and their results can be found in Appendix A.

5. Query Evaluation Strategy

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 6.3) and graph evolution metrics (Section 6.4) and evaluating pivot query expressions (Section 6.5).

An alternative strategy would be to materialize pivot graphs during query processing. This would avoid data preprocessing and data storage costs but also lead to higher query execution time. Whereas we believe the first strategy is better adapted to our current query-based graph exploration application, the second scenario is part of future work. Our goal is in particular to study optimization strategies which exploit the commutativitiy and monotonicity of filter predicates to generate efficient query execution plans.

5.1. EPIQUE Workflow

Figure 3 and Algorithm 1 detail the main steps of the EPIQUE workflow.

Algorithm 1 Workflow Overview

```
1:  procedure GeneratePivotGraphs(C, P, B)
2:      C' ← preprocessing(C)
3:      C'' ← periodization(C',P)
4:      T ← topic_modeling(C'')
5:      G ← topic_alignment(T)
6:      A ← compute_pivot_graphs(G, B)
7:      L ← compute_labels(A)
8:      M ← compute_graph_metrics(A)
9:      return (A,L,M)
10: end procedure
```

The input of the workflow is a document corpus C, a set of possibly overlapping time periods P, a list of β-thresholds B and some method-specific parameters (like the number of topics for LDA). The workflow starts with a standard document pre-processing step composed of some lexical analysis, stop-word removal, stemming, index term generation and term selection. The main goal of this step is to extract for each document a term vector which precisely describes the scientific document contents.

The document preprocessing step is followed by a corpus periodization step which decomposes the document collection C' according to a sequence of continuous and possibly overlapping time periods P (left part of Figure 3). Each *period* defines the subset of documents published during the period. The choice of the window size and sliding step depends on the granularity of the document time-stamps (year, month, day) and on the number of available documents in each period. In the following step, each periodized corpus in C'' is analyzed by a topic model. In our prototype, we use the LDA [8] implementation provided by the machine learning library Spark MLlib [27] to extract the topics for each corpus period. The output of LDA is a topic-term matrix describing each topic as a weighted term vector. LDA requires to set the number of topics to be generated in advance. Tuning this parameter is important and subtle because it strongly influences the diversity of the topics generated for each time period. We show in Section 7.2 how experts can be assisted in choosing the right number of topics.

The extracted topic-term vectors T from different periods are aligned with an appropriate similarity measure to produce a complete alignment graph Sim. The complete alignment graph Sim connecting all topics from different periods is necessary to compute the pivot evolution graphs as shown in Section 5.2. It

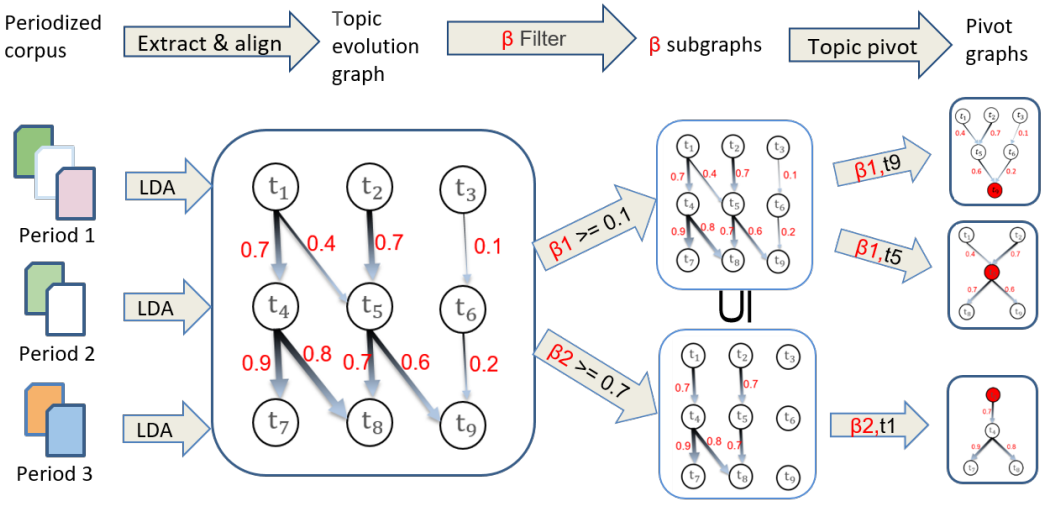


Figure 3: Topic evolution model of EPIQUE workflow

also can be used to extract the topic evolution graph G (central part of Figure 3) connecting all topics from subsequent periods. In our experiments, we use cosine similarity which performs well for aligning sparse topic-term vectors. Observe that the choice of LDA and cosine similarity does not exclude the use of other topic models and similarity measures like Jaccard similarity [19] or Battacharya similarity [6][11].

The next step produces the pivot evolution graphs following the model introduced in Section 3 (right part of Figure 3). The global evolution graph G is transformed into $|B|$ families of pivot evolution graphs defined by a set of alignment thresholds $\beta_i \in B$. Each family contains the future and past pivot graphs of all pivot topics (t, β_i). We only consider pivot graphs with at least one edge and ignore isolated pivot topics (single node graphs). The final database then contains at most $|B| \times |T|$ pivot topics with a future and a past pivot evolution graph for pivot topic.

The last two steps compute the topic labels and graph metrics as defined in Section 4. The resulting data can then be queried using the filters defined in Section 4.3. In the following section, we describe in more detail the central task of this workflow concerning the computation of the pivot topic graphs. An implementation of the workflow is described in Section 6.

5.2. Pivot Graph Computation

We use a Datalog-like syntax for presenting our pivot graph generation approach. An efficient algorithm and implementation on top of Apache Spark is discussed in Section 6. This implementation 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 $0 \leq i < n$.

The input of the Datalog program (Algorithm 2) is a topic table $Topics(t, p)$ storing the pivot topic identifiers t with their periods p, a topic similarity matrix $Sim(t1, t2, s)$ where $s = sim(t1, t2)$ 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 (Rule 1) defines the alignment graph table $Graph(t1, t2, s)$ which connects all topics $t1$ of period p to all topics $t2$ of period $p + 1$ where there exists an evolution edge of similarity $s = sim(t1, t2)$. Starting from $Graph$, the following rules (Rules 2 to 7) compute all pivot topic evolution graphs for all topics in T and beta value β_i. This is done by first generating table $TC(t1, t2, d, i)$ containing the transitive closure of the alignment subgraph of $Graph$ where all edges have a weight greater or equal to β_i and d is the graph distance

[11] In [29] we have addressed the problem of computing very large cosine-similarity graphs in parallel on Apache Spark.

between $t1$ and $t2$[12]. Tables $Past$ and $Future$ contain the pivot topic evolution graphs $\mathcal{G}^{\delta}(t, \beta_i)$ of all topics t where a tuple (t, x, y, rs, ps, d, i) represents an edge (x, y, s) in pivot graph $\mathcal{G}^{\delta}(t, \beta_i)$ of pivot topic t 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.

Algorithm 2 Pivot graph computation using Datalog

```
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).
```

Table size estimation: We can estimate the size of each table 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. Then the size (number of edges) of $Graph$ is bound by $|Graph| \leq k * t * (p-1)$: $Graph$ is a (directed acyclic) multistage graph where each topic (except the leaves) have maximally k child links. 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 7) and current big data frameworks can easily manage graphs of this size.

6. Implementation

This section presents the implementation of the topic evolution model in the context of the EPIQUE project.

6.1. System Architecture

Figure 4 gives an overview of the architecture of our web application implemented on top of Apache Spark and Jupyter Notebook. The entire process to study science evolution over a corpus is split into two applications for building the pivot evolution graphs and for interactively exploring these graphs. Each application corresponds to a separate user interface. The evolution graph generation application is implemented in Scala and executed through the Spylon[13] kernel. The evolution graph exploration application uses a standard Python kernel to take advantage of advanced Python 3 graphical user interface libraries for facilitating user interaction.

Figure 5 displays a screenshot of our intuitive declarative query-by-example interface where users can specify their exploration goal and visualize pivot topic evolution graphs. These graphs are pre-computed in order to ensure fast query answer display.

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

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

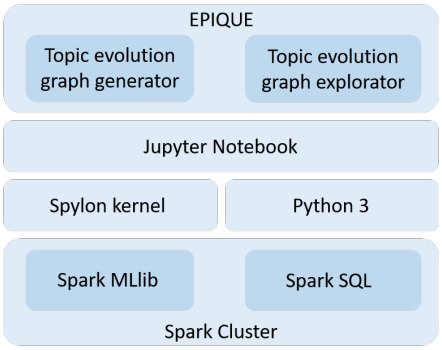


Figure 4: Architecture overview of EPIQUE web application

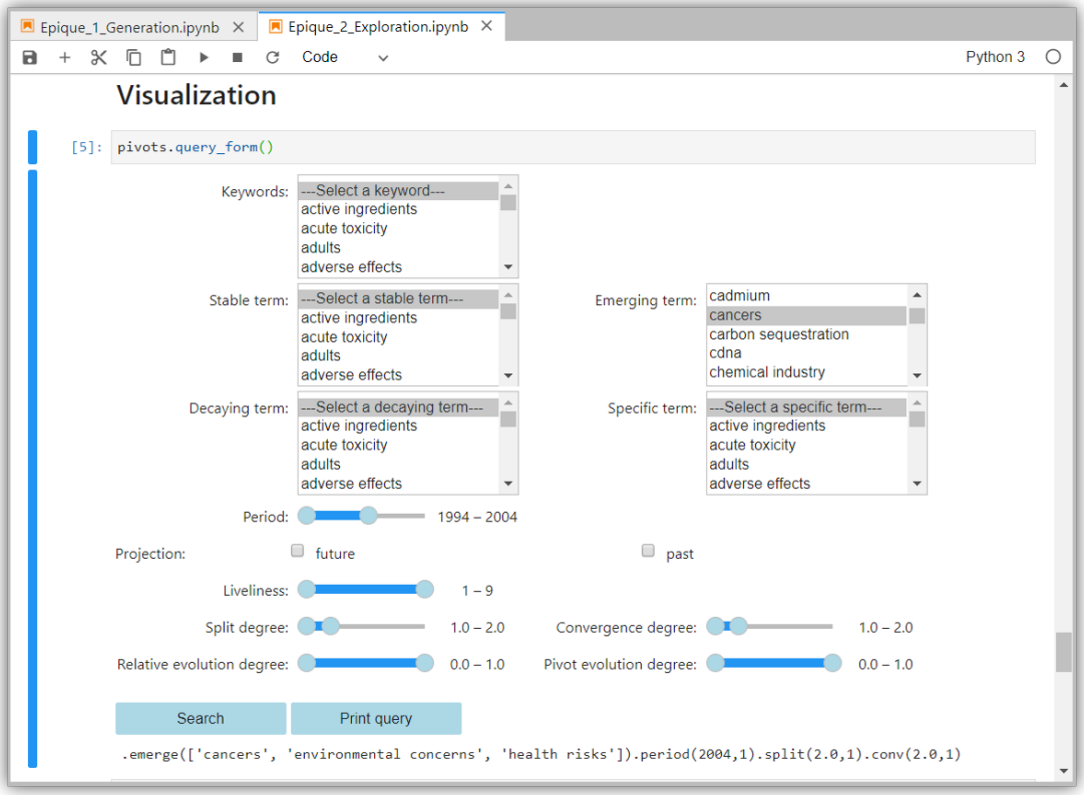


Figure 5: Screenshot of the UI of EPIQUE web application

6.2. Pivot graph generation algorithm

Algorithm 3 on Page 17 shows our pivot graph computation implementation using the Spark DataFrame [2] model and algebra. A Spark DataFrame is equivalent to a relational database table and can be transformed into a new DataFrames 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

Algorithm 3 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 β_i.
for $i = n...0$ **do**
    // Graph(t_1, t_2, sim) : topic alignment graph for β_i (rule 1 in Algorithm 2)
    Graph = Sim.where($sim \geqslant \beta_i \wedge t_1.period + 1 = t_2.period$)
    // Transitive closure computation (rules 3 and 4 in Algorithm 2)
    if i=n **then**
        oldTC = $\emptyset$ // *oldTC is empty for* β_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 2)
    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 2)
    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 2)
    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* β_i
    oldTC=newTC
end for

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 3 computes and stores for each alignment threshold β_i $(0 \leq i \leq n)$ two tables $Future$ and $Past$ as defined in Section 5.2. 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 5.2. 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 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 [3] 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 2). Both tables are stored at the end of each iteration step.

6.3. Topic labeling

Pivot topic labels can be computed with Spark SQL [2] by defining a query over the pivot evolution tables $Future(t, x, y, rs, ps, d, i)$ and $Past(t, x, y, rs, ps, d, i)$ as defined in Section 5.2. 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 4.2. The final query on view *TmpLabels* uses three set operations *group_intersect*, *group_except* and *group_union* to compute the stable, emerging, decaying and specific terms as defined in Section 4.1. The result is stored in a new table *PivotLabels*.

Listing 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
)
```

6.4. Graph Metrics Computation

The liveliness, relative evolution degree, pivot evolution degree, split degree and convergence degree of all pivot evolution graphs can directly be computed by a standard SQL aggregation query. The following query computes these metrics for all $Future(t, x, y, rs, ps, d, i)$ pivot evolution graphs. Due to the precomputation of pivot graphs, more graph metrics can be integrated flexibly and carried out immediately by using this simple SQL aggregation query in the future:

Listing 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
```

Table 3
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

```
group by t, i
```

6.5. Pivot Queries

Pivot queries are translated into standard SQL query where each pivot filter becomes a predicate in the query where clause:

Listing 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
```

Observe that for evaluating the pivot query filters presented in Section 4.3 without the *Path*-operator, it is sufficient to store *Graph* (for visualization), *PivotFuture*, *PivotPast*, *PivotAll* (for filtering) and *PivotLabels*. An efficient implementation of *Path* queries, for example by using graph-labeling schemes [37] for checking node reachability in acyclic graphs, is part of our future work.

7. Experiments

7.1. Experiment Setting

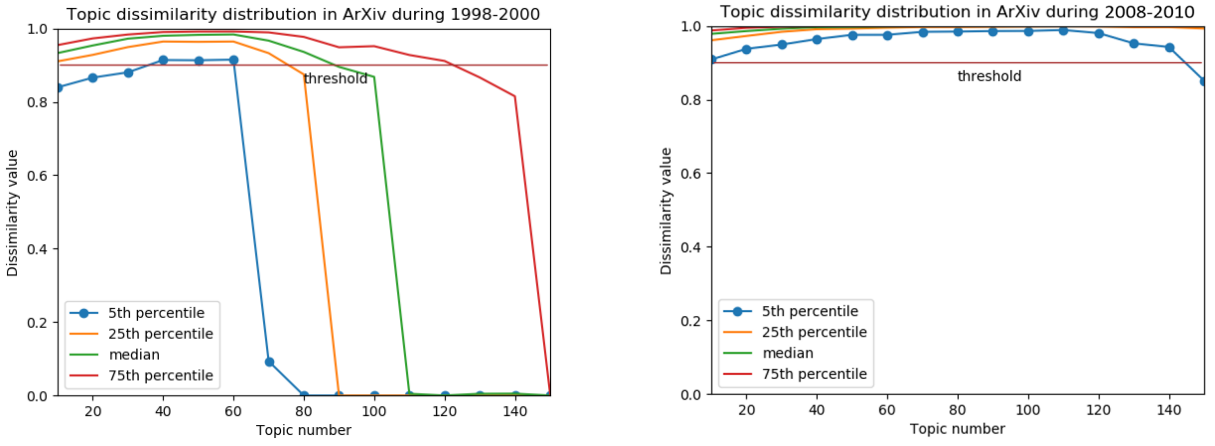
We conducted our experiments on four real-world data sets of different scales by using the titles and the abstracts of each document. The smallest dataset contains 4640 documents about research related to the Glyphosate herbicide. The second dataset ISTEX contains 13 423 articles in the domain of ecological 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 over these four data sets are summarized in Table 3, where #*Document* is the total number of documents and #*T* is the number of topics per period we used in our experiments. All datasets cover a different period of 20 years which is split into 10 slices 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. In particular, we did not apply any sampling before generating the document-term matrix and executing LDA topic extraction.

The EPIQUE workflow is implemented on top of Apache Spark version 2.4, Scala version 2.11 and Java version 8. Our experiments have shown that most workflow stages can be efficiently executed on a single node. 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. For testing the scalability of the pivot computation algorithm (Section 7.4) we use a Spark cluster with 11 nodes.

7.2. Diversity-based Topic Number Selection



(a) 1998-2000 **(b)** 2008-2010

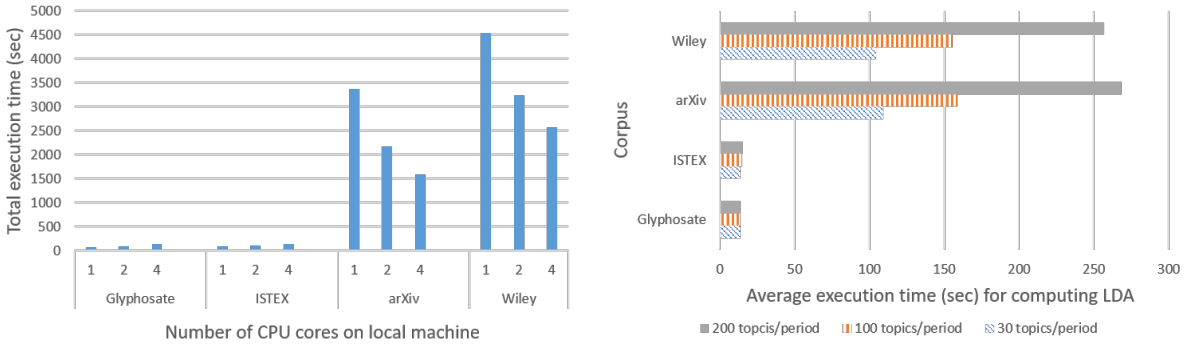
Figure 6: Dissimilarity distribution (diversity) by number of topics in arXiv.CS

In order to build pivot graphs over more representative topic sets, we use *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. For example, Figure 6(a) and Figure 6(b) 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 from 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 6(a) that for a topic number $\#T$ ranging between 40 and 60, less than 5 percent (blue line) of all topic pairs have a similarity value higher than 0.1 (dissimilarity value lower than 0.9) and any number in this range is a good choice. Figure 6(b) shows that for the larger corpus, LDA achieves high diversity even for 140 topics. 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). 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.

7.3. LDA Topic Computation

LDA is applied on a document-term matrix containing the term frequency for each term/document pair. This matrix has been 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 7(a) 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



(a) LDA performance *wrt.* number of CPU cores **(b)** Average execution time for computing LDA VS. $\#T$

Figure 7: LDA performance evaluation on local machine

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 7(b) presents 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 computation for LDA in our workflow is only done once and the performance 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.

7.4. Pivot Graph Computation Performance

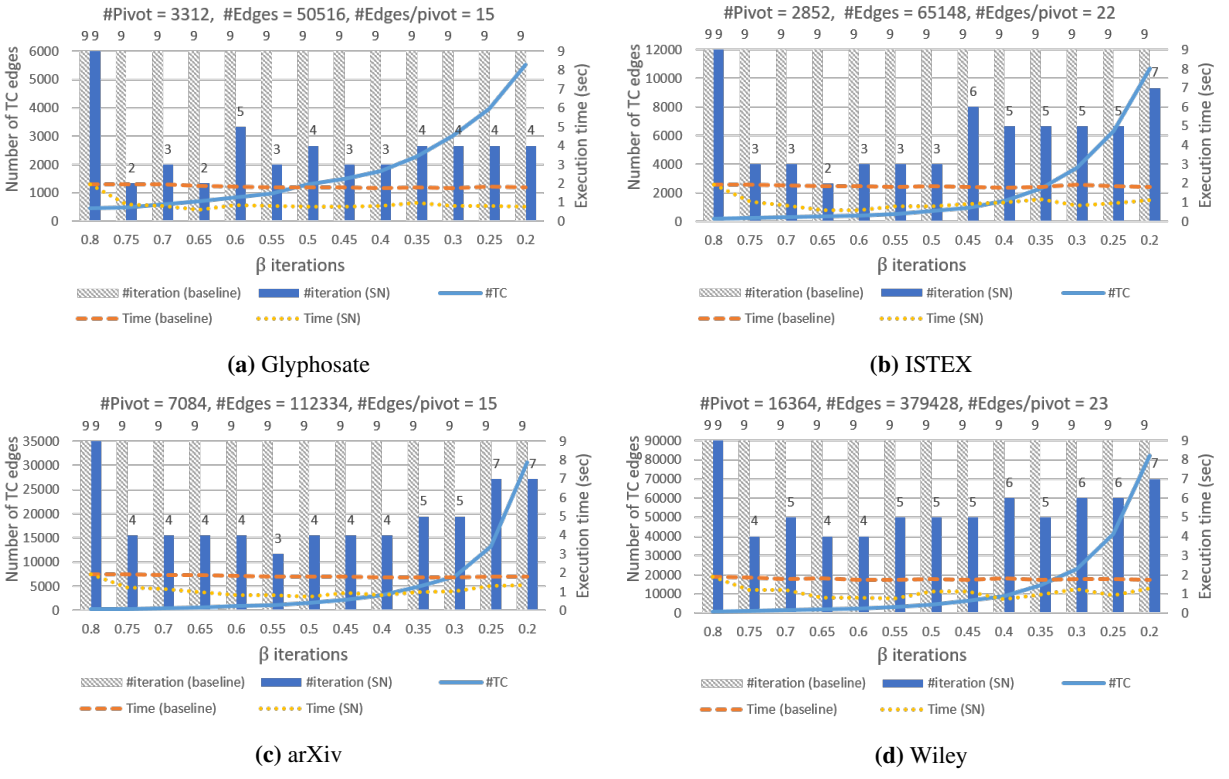
The generation of pivot graphs is the main computation step of our workflow including many join queries necessary for the transitive closure computations (TC graph) of pivot graphs for different β values.

We compare two algorithms. The baseline algorithm (baseline) computes for each β threshold, the pivot graphs of all corresponding pivot topics from scratch whereas the semi-naive Algorithm 3 (SN) incrementally reuses the result of previous iterations to compute the pivot graphs of new topics for a lower β value.

Figure 8 gives a deeper insight of these costs and displays for each corpus, the transitive closure (TC) computation time by the baseline algorithm (orange line) and the optimized semi-naive Algorithm 3 (yellow line). The β values span from 0.2 to 0.8. The baseline algorithm (gray bars) applies for each β threshold 9 iterations (join queries) for generating the TC edges over all 10 periods, whereas the number of joins in the semi-naive 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 baseline algorithm (orange dashed line) and the execution time of the semi-naive algorithm (dotted dashed line) is reduced by about 50%. We also can observe that, even if the size of the joined table increases exponentially (blue solid line), the execution times 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 semi-naive evaluation (SN), we have conducted 3 groups of experiments for each corpus with β intervals ranging from 0.2 to 0.8 of different granularity. The results are shown in Figure 9. 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 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 semi-naive (SN) algorithm outperforms the baseline solution.

As shown in Figure 8, our optimized algorithm is capable of computing efficiently the pivot graphs for



(a) Glyphosate **(b)** ISTEX

(c) arXiv **(d)** Wiley

Figure 8: Correlation between the execution time and the number of iterations of transitive closure computation for each β iteration

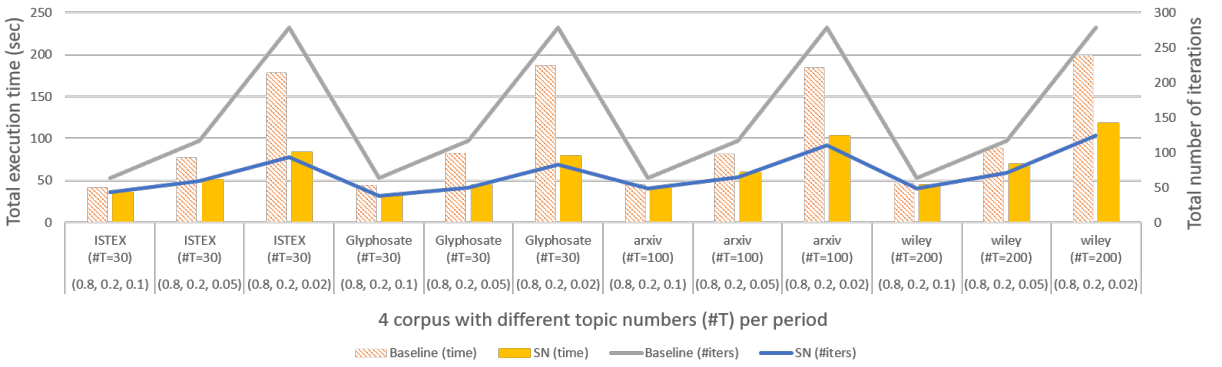
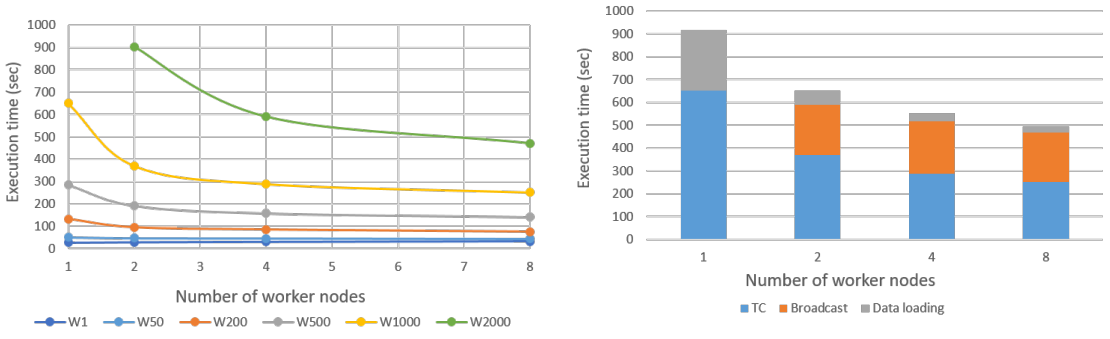


Figure 9: Comparison between the baseline implementation and the optimized semi-naïve implementation

all real-world topic graphs on a single machine. The largest TC table is computed from the Wiley topic graph which contains 2000 topics and 360 000 alignment edges and generates 16 364 pivot graphs with 379 428 edges (tuples). Our experiments also showed 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), 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 in-

cluding the document storage, preprocessing (stop-word removal, stemming, term extraction) steps which can benefit of a distributed Spark architecture.



(a) Performance evaluation with synthetic datasets

(b) Execution time of different steps to compute TC for W1000 on the cluster VS. number of worker nodes

Figure 10: Scalability of TC computation with synthetic datasets

For illustrating the pivot graph 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 fig. 10(a). Each line corresponds to the execution 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 aforementioned topic graph 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.

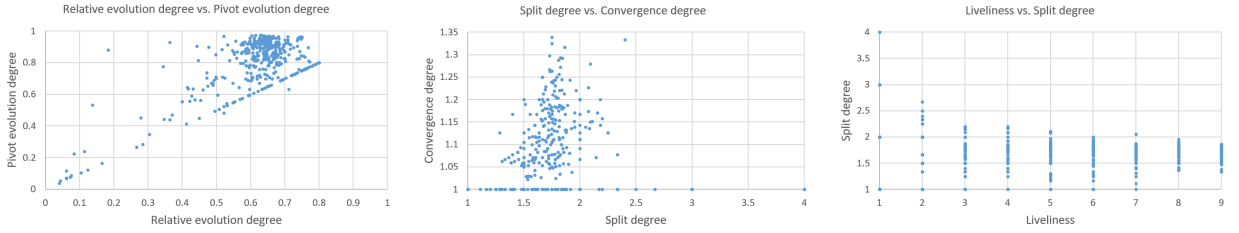
The number of physical worker nodes ranges from 1 to 8 and each worker node contains one Spark executor configured with 4 CPU cores and 40 GB memory. Figure 10(a) illustrates the TC computation time for the synthetic topic graphs $W1$, $W50$ $W200$, $W500$, $W1000$ and $W2000$. The edges of all graphs 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 10(b) shows the total execution time according to the different TC computation steps applied on $W1000$ with one up to 8 worker nodes. The blue bars correspond to the iterative TC compution costs shown in Figure 10(a) (yellow line). For example, the TC computation time on one node in Figure 10(b) is 650 seconds which corresponds to the point of the yellow line for one node in Figure 10(a). We apply the distributed broadcast join algorithm implemented in Spark SQL[14]. 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 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).

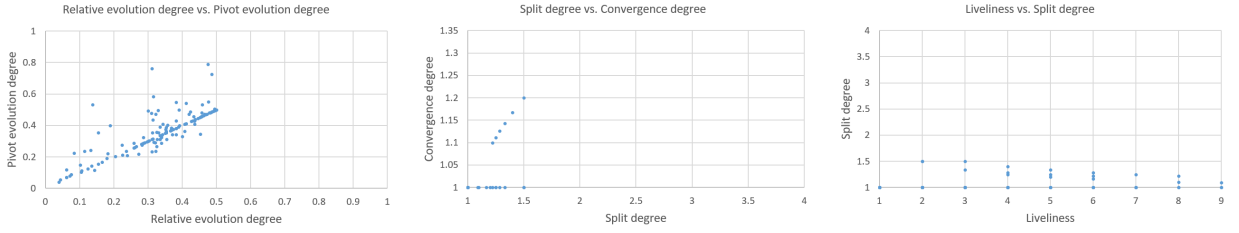
[14]https://mungingdata.com/apache-spark/broadcast-joins/

7.5. Pivot Topic Analysis

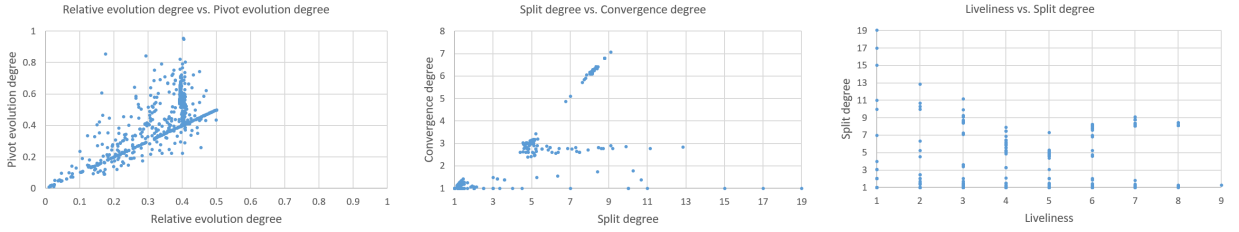
The metrics defined in Section 4.1 can be used for the structural and quantitative analysis of the evolution of topics. The objective of this section is to explore the impact of the main parameters, *i.e.*, the β threshold and the topic number $\#T$, on the structure and the semantics of the generated pivot evolution graphs.



(a) $\beta = 0.2$, $\#T = 50$, #Pivot = 477, #Isolated = 23 **(b)** $\beta = 0.2$, $\#T = 50$, #Pivot = 477, #Isolated = 23 **(c)** $\beta = 0.2$, $\#T = 50$, #Pivot = 477, #Isolated = 23



(d) $\beta = 0.5$, $\#T = 50$, #Pivot = 198, #Isolated = 302 **(e)** $\beta = 0.5$, $\#T = 50$, #Pivot = 198, #Isolated = 302 **(f)** $\beta = 0.5$, $\#T = 50$, #Pivot = 198, #Isolated = 302



(g) $\beta = 0.5$, $\#T = 150$, #Pivot = 1014, #Isolated = 486 **(h)** $\beta = 0.5$, $\#T = 150$, #Pivot = 1014, #Isolated = 486 **(i)** $\beta = 0.5$, $\#T = 150$, #Pivot = 1014, #Isolated = 486

Figure 11: Distribution of future pivot evolution graphs in arXiv with respect to three groups of metrics by varying β and $\#T$.

Figure 11 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 11(a) with Figure 11(d), 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 pivot topics to connect with more topics than higher β values which only connect similar topics. This is shown in Figure 11(b) which represents a large number of complex pivot topic graphs with higher split and convergence degrees

than the pivot topic graphs in Figure 11(e). The previous observation is also confirmed in Figure 11(c) and Figure 11(f) 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 11(g), pivot topics tend to evolve a lot even for a low relative evolution degree. The pivot graphs in Figure 11(h) are much more complex than the graphs generated by the same β-threshold with $\#T = 50$ topics (Figure 11(e) and Figure 11(f)). As we can see, the *split degree* attains a value of 19 compared with maximal *split degree* 1.5 in Figure 11(e). 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.

Our query language allows users to select topics in specific regions of the sub-figures in Figure 11. For example query $Q1$ in Example 5 chooses all topics which appear in the upper right window of Figures 11(a) and on the right part of Figure 11(b) on the line corresponding to the liveliness value 5 in Figure 11(c).

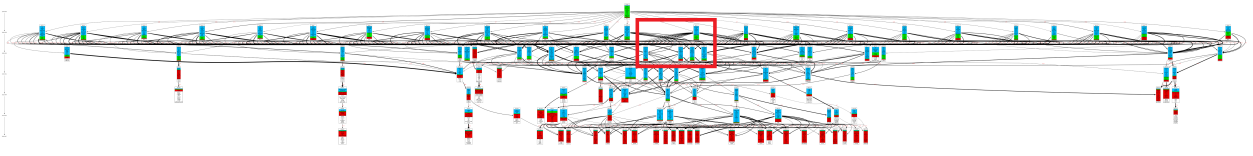


Figure 12: Visualization of pivot graph $\mathcal{G}^{future}(495, 0.5)$ where $\#T = 150$

Figure 12 shows a future arXiv pivot graph generated for $\#T = 150$ and $\beta = 0.5$, which corresponds to a data point in Figure 11(h) 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 13 zooms into the rectangle of Figure 12 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 7.2 which reduces the number of documents per period).

8. Conclusion and Future Work

We have presented a new framework for the visualisation and exploration of topic evolution networks representing the progress and evolution of research in scientific document archives. This framework is based on the pivot graph model which represents the evolution of each topic by a set of connected topic evolution subgraphs. Using this model, the user can express complex filter queries to obtain the relevant pivot topic graphs. The model has been implemented on top of Apache Spark using LDA and for topic extraction and Spark SQL for computing pivot topic graphs. We also proposed an efficient incremental transitive closure algorithm which reduces the number of SQL joins. A first prototype [24] is currently used to extract and analyze evolution patterns for different scientific domains in collaboration with philosophers of science and as part of the EPIQUE project.

As future work we intend to explore an alternative pivot graph materialization strategy based on GraphX [40]. The current relational implementation is mainly based on efficient distributed map-reduce algebra of Spark SQL. However, as we have seen in our experiments, the transitive closure computation generates many join query tasks. Whereas each join is executed very rapidly, the task scheduling overhead is important and the current implementation cannot completely benefit from the parallel Spark architecture.

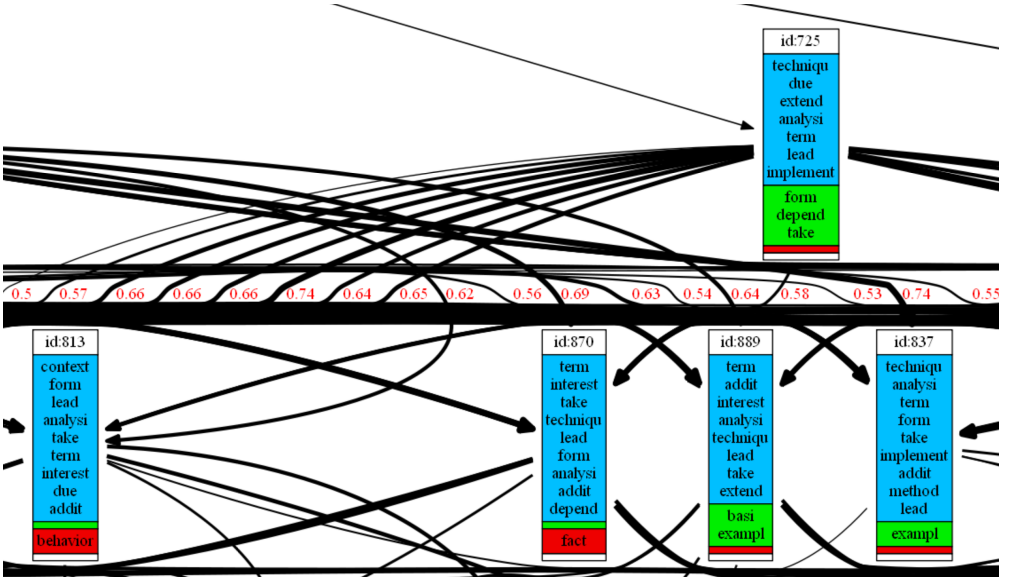


Figure 13: Zoom in Figure 12

Therefore, we intend to apply the Pregel operator of GraphX which implements the bulk-synchronous parallel (BSP) messaging abstraction for computing pivot topic labels and statistics. In parallel, we study the implementation of pivot filters over the original topic evolution graphs (without a preliminary materialization step) and the opportunities to exploit the monotonicity of certain pivot filters for optimizing complex queries by cost-based query rewriting strategies.

References

[1] Andrei, V., Arandjelović, O., 2016. Complex temporal topic evolution modelling using the Kullback-Leibler divergence and the Bhattacharyya distance. EURASIP Journal on Bioinformatics and Systems Biology 2016, 16.

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

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

[4] Beykikhoshk, A., Arandjelović, O., Phung, D., Venkatesh, S., 2018. Discovering topic structures of a temporally evolving document corpus. Knowledge and Information Systems 55, 599–632.

[5] Bhadury, A., Chen, J., Zhu, J., Liu, S., 2016. Scaling Up Dynamic Topic Models, in: Proceedings of the 25th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland. pp. 381–390.

[6] Bhattacharyya, A., 1943. On a measure of divergence between two statistical populations defined by their probability distributions. Bulletin of the Calcutta Mathematical Society 35, 99–109.

[7] Blei, D.M., Lafferty, J.D., 2006. Dynamic Topic Models, in: Proceedings of the 23rd International Conference on Machine Learning, ACM, New York, NY, USA. pp. 113–120.

[8] Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent dirichlet allocation. Journal of machine Learning research 3, 993–1022.

[9] Chavalarias, D., Cointet, J.P.P., 2013. Phylomemetic patterns in science evolution—the rise and fall of scientific fields. PloS one 8, e54847.

[10] Chen, B., Tsutsui, S., Ding, Y., Ma, F., 2017. Understanding the topic evolution in a scientific domain: An exploratory study for the field of information retrieval. Journal of Informetrics 11, 1175–1189.

[11] Cohen Priva, U., Austerweil, J.L., 2015. Analyzing the history of Cognition using Topic Models. Cognition 135, 4–9.

[12] Franz, M., Ward, T., McCarley, J.S., Zhu, W.J., 2001. Unsupervised and Supervised Clustering for Topic Tracking, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA. pp. 310–317.

[13] Garfield, E., 1955. Citation Indexes for Science: A New Dimension in Documentation through Association of Ideas. Science 122, 108–111.

[14] Griffiths, T.L., Steyvers, M., 2004. Finding scientific topics. Proceedings of the National Academy of Sciences 101, 5228–5235.

[15] Hall, D., Jurafsky, D., Manning, C.D., 2008. Studying the History of Ideas Using Topic Models, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Stroudsburg, PA, USA. pp. 363–371.

[16] He, Q., Chen, B., Pei, J., Qiu, B., Mitra, P., Giles, L., 2009. Detecting Topic Evolution in Scientific Literature: How Can Citations Help?, in: ACM Conf. on Information and Knowledge Management, New York, NY, USA. pp. 957–966.
[17] Hofmann, T., 1999. Probabilistic latent semantic indexing, in: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 50–57.
[18] Hu, B., Dong, X., Zhang, C., Bowman, T.D., Ding, Y., Milojević, S., Ni, C., Yan, E., Larivière, V., 2015. A Lead-lag Analysis of the Topic Evolution Patterns for Preprints and Publications. J. Assoc. Inf. Sci. Technol. 66, 2643–2656.
[19] Jaccard, P., 1912. The Distribution of the Flora in the Alpine Zone.1. New Phytologist 11, 37–50.
[20] Jo, Y., Hopcroft, J.E., Lagoze, C., 2011. The web of topics: discovering the topology of topic evolution in a corpus, in: Proceedings of the 20th international conference on World wide web, ACM. pp. 257–266.
[21] Kontostathis, A., Galitsky, L.M., Pottenger, W.M., Roy, S., Phelps, D.J., 2004. A survey of emerging trend detection in textual data mining, in: Survey of text mining. Springer, pp. 185–224.
[22] Kuhn, T.S., Neurath, O., Kuhn, T.S., 1994. The Structure of scientific revolutions. Number ed.-in-chief: Otto Neurath ; Vol. 2 No. 2 in International encyclopedia of unified science Foundations of the unity of science. 2nd ed., enlarged ed., Chicago Univ. Press, Chicago, Ill. OCLC: 258260085.
[23] Kullback, S., Leibler, R.A., 1951. On information and sufficiency. Ann. Math. Statist. 22, 79–86.
[24] Li, K., Naacke, H., Amann, B., 2020a. Epique: Extracting meaningful science evolution patterns from large document archives, in: International Conference on Extending Database Technology (EDBT), Copenhagen, Denmark. pp. 619–522. Demonstration.
[25] Li, K., Naacke, H., Amann, B., 2020b. Exploring the evolution of science with pivot topic graphs, in: International Workshop on Big Data Visual Exploration and Analytics BigVis (EDBT 2020), pp. 1–8.
[26] Lu, B., Ott, M., Cardie, C., Tsou, B.K., 2011. Multi-aspect sentiment analysis with topic models, in: 2011 IEEE 11th international conference on data mining workshops, IEEE. pp. 81–88.
[27] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., others, 2016. Mllib: Machine learning in apache spark. The Journal of Machine Learning Research 17, 1235–1241.
[28] Miller, J.W., Harrison, M.T., 2013. A simple example of dirichlet process mixture inconsistency for the number of components, in: Advances in neural information processing systems, pp. 199–206.
[29] Naacke, H., Ke, L., Amann, B., Curé, O., 2019. Efficient similarity-based alignment of temporally-situated graph nodes with apache spark, in: 2019 IEEE International Conference on Big Data (Big Data), IEEE. pp. 4793–4798.
[30] Niu, Z., Hua, G., Wang, L., Gao, X., 2017. Knowledge-based topic model for unsupervised object discovery and localization. IEEE Transactions on Image Processing 27, 50–63.
[31] Rubin, T.N., Chambers, A., Smyth, P., Steyvers, M., 2012. Statistical topic models for multi-label document classification. Machine learning 88, 157–208.
[32] Salatino, A.A., Osborne, F., Motta, E., 2018. AUGUR: Forecasting the Emergence of New Research Topics, in: ACM/IEEE on Joint Conference on Digital Libraries, ACM, New York, NY, USA. pp. 303–312.
[33] Shahaf, D., Guestrin, C., Horvitz, E., Leskovec, J., 2015. Information cartography. Commun. ACM 58, 62–73.
[34] Sun, X., Kaur, J., Milojević, S., Flammini, A., Menczer, F., 2013. Social Dynamics of Science. Scientific Reports 3, 1069.
[35] Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M., 2005. Sharing clusters among related groups: Hierarchical Dirichlet processes, in: Advances in neural information processing systems, pp. 1385–1392.
[36] Wang, C., Blei, D., Heckerman, D., 2008. Continuous Time Dynamic Topic Models, in: Conference on Uncertainty in Artificial Intelligence, AUAI Press, Arlington, Virginia, United States. pp. 579–586.
[37] Wang, H., He, H., Yang, J., Yu, P.S., Yu, J.X., 2006. Dual labeling: Answering graph reachability queries in constant time, in: 22nd International Conference on Data Engineering (ICDE'06), IEEE. pp. 75–75.
[38] Wang, X., McCallum, A., 2006. Topics over Time: A non-Markov Continuous-time Model of Topical Trends, in: Int'l Conf. on Knowledge Discovery and Data Mining, ACM, New York, NY, USA. pp. 424–433.
[39] Wheeler, D.L., Barrett, T., Benson, D.A., Bryant, S.H., Canese, K., Chetvernin, V., Church, D.M., DiCuccio, M., Edgar, R., Federhen, S., et al., 2007. Database resources of the national center for biotechnology information. Nucleic acids research 36, D13–D21.
[40] Xin, R.S., Gonzalez, J.E., Franklin, M.J., Stoica, I., 2013. Graphx: A resilient distributed graph system on spark, in: First international workshop on graph data management experiences and systems, pp. 1–6.
[41] Zhou, D., Ji, X., Zha, H., Giles, C.L., 2006. Topic Evolution and Social Interactions: How Authors Effect Research, in: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, ACM, New York, NY, USA. pp. 248–257.
[42] Zuo, Z., Zhao, K., 2018. A Graphical Model for Topical Impact over Time, in: Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, ACM, New York, NY, USA. pp. 405–406.

A. Appendix

In the following examples, all pivot topics are located at the first period in each graph. Observe that the user does not specify the β-threshold. Although Figure 14 and Figure 16 have the same structure, they have different evolution pace (corresponding to different β values). The pivot graph in Figure 14 has more emerging terms (green part) whereas the pivot graph in Figure 16 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 212 (Figure 16), pivot topic 331 (Figure 15) has a shorter future and a little bit

more complex evolution structure.

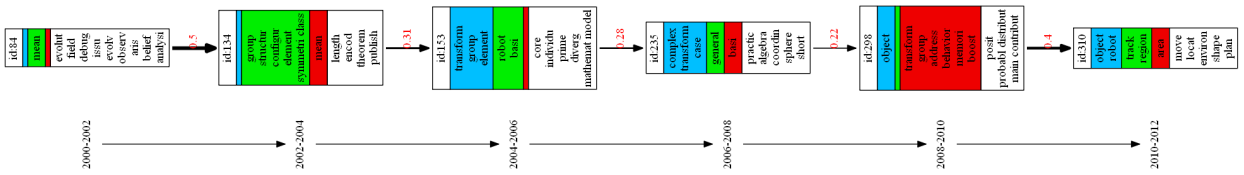


Figure 14: $\overline{Q4}$: $revol^{future} \geq 0.5 \wedge pevol^{future} \geq 0.6 \wedge split^{future} \leq 1.2 \wedge live^{future} = 5$

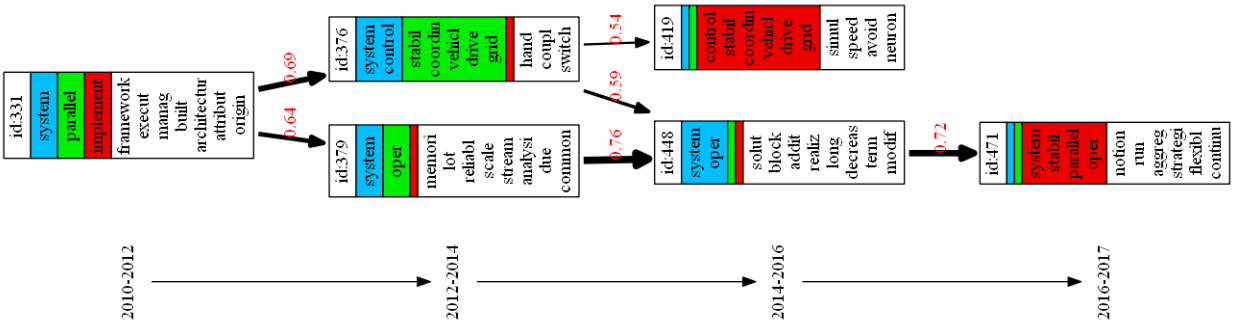


Figure 15: $\overline{Q5}$: $revol^{future} \leq 0.4 \wedge pevol^{future} \leq 0.5 \wedge split^{future} \geq 1.5 \wedge live^{future} = 3$

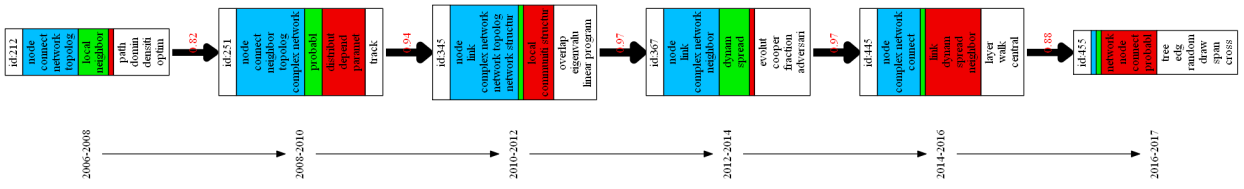


Figure 16: $\overline{Q6}$: $revol^{future} \leq 0.4 \wedge pevol^{future} \leq 0.5 \wedge split^{future} \leq 1.2 \wedge live^{future} = 5$