MINING LATENT ENTITY STRUCTURES FROM MASSIVE UNSTRUCTURED AND INTERCONNECTED DATA

BY

CHI WANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Abstract

The "big data" era is characterized by an explosion of information in the form of digital data collections, ranging from scientific knowledge, to social media, news, and everyone's daily life. Valuable knowledge about multi-typed entities is often hidden in the unstructured or loosely structured but interconnected data. Mining latent structured information around entities uncovers semantic structures from massive unstructured data and hence enables many high-impact applications, including taxonomy or knowledge base construction, multi-dimensional data analysis and information or social network analysis.

A mining framework is proposed, to solve and integrate a chain of tasks: hierarchical topic discovery, topical phrase mining, entity role analysis and entity relation mining. It reveals two main forms of structures: topical and relational structures. The topical structure summarizes the topics associated with entities with various granularity, such as the research areas in computer science. The framework enables recursive construction of phrase-represented and entity-enriched topic hierarchy from text-attached information networks. It makes breakthrough in terms of quality and computational efficiency. The relational structure recovers the hidden relationship among entities, such as advisor-advisee. A probabilistic graphical modeling approach is proposed. The method can utilize heterogeneous attributes and links to capture all kinds of semantic signals, including constraints and dependencies, to recover the hierarchical relationship with the best known accuracy.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

The success of database technology is largely attributed to the efficient and effective management of *structured data*. The construction of a well-structured database is often the premise of subsequent applications. However, the explosion of "big data" poses great challenges on this practice since the real world data are largely unstructured, or loosely structured. Thus, it is crucial to uncover latent structures of real-world entities, such as people, locations and organizations, and bring massive unstructured data into better semantic structures. By mining massive unstructured or loosely structured data where the entities occur, one can construct semantically rich structures which reveal the relationships among entities and provide topical grouping of them. The uncovered entity relationships and groupings facilitate browsing information and retrieving knowledge from data that are otherwise hard to handle due to the lack of structures.

Example: latent entity structures in scientific literature and news media. In a bibliographic database like DBLP[1] and PubMed[2], research papers are explicitly linked with authors, venues and terms. Many interesting semantic relationships such as advisor-advisee between authors are hidden in the publication records; moreover, the research topics of authors, venues and terms are also hidden or unorganized, preventing insightful organization of the entities. In news articles, multiple types of entities like people, locations and organizations are involved in many events of different topics. The topics, as well as the entity relations are buried in the text rather than in the form of relational tuples.

Mining latent entity structures helps with many knowledge engineering tasks and services. In

[1] `http://www.informatik.uni-trier.de/`
[2] `http://www.ncbi.nlm.nih.gov/pubmed/`

the web scale, it is crucial to mine the entity structures hidden in web pages and tables since they are extensive resources to enrich open-domain knowledge-bases; also, mining the entity structures hihdden in social media will help reorganize scattered information and improve the service for individuals.

1.2 Data model

This section introduces a model for unstructured and interconnected data. We notice that many datasets nowadays contain both unstructured part and weak structures, as the followinge examples illustrate.

Example 1.1 (Research publication) *A bibliographic database like DBLP and PubMed contains unstructured paper text (titles, abstracts, full text etc.), as well as structures, in the sense that every paper is explicitly linked with authors, venues, and years.*

Example 1.2 (News article) *A collection of news articles contains unstructured news text (titles, snippets, full text etc.), as well as weak structures, in the sense that every news article is implicitly linked with named entities like persons, locations and organizations.*

Example 1.3 (Social media) *A social media website like Twitter contains unstructured tweets, as well as structures, in the sense that every tweet is linked with twitters, urls or hashtags. There are also links among the twitters by their following relationship. In addition, every twitter is linked to his/her profile. The profile may have structured fields such as interests and locations, while each field may contain unstructured text.*

This thesis studies a data model that is a abstract of these examples.

Definition 1 (Text-attached heterogeneous information network) *A text-attached heterogeneous information network can be formally represented by a tuple $H = (\{\mathcal{V}_x\}, \{\mathcal{E}_{x,y}\}, \mathcal{D})$. $\mathcal{V}_x$ is the set of type-x nodes, and $\mathcal{E}_{x,y}$ is the set of link weights between type x and type y nodes (x and y may be identical). $\mathcal{D} = \{d_v\}$ is the set of documents, where d_v is the document attached to node v. d_v can be empty if node v is not attached with text.*

The research publication dataset, for example, can be formally represented as follows.

Example 1.4 (DBLP network) *There are three types of nodes: $\mathcal{V}_1$ for papers, $\mathcal{V}_2$ for authors, and $\mathcal{V}_3$ for venues. There are two types of links: $\mathcal{E}_{1,2}$ for the links between papers and authors, implying a paper is written by an author; and $\mathcal{E}_{1,3}$ for the links between papers and venues, implying a paper is published in a venue. There is one document d_i attached to every paper node $v_i^1 \in \mathcal{V}_1$, i.e., the title of the paper. Note that only the titles are available as unstructured text in the DBLP database.*

Our data model is the generalization of several simpler forms of data: i) text-only corpus, in which $\mathcal{E} = \emptyset$ and $|\mathcal{D}| = |\mathcal{V}|$; ii) text-absent heterogeneous network, in which $\mathcal{D} = \emptyset$; and iii) textual corpus with meta data or star-schema network, in which all the links share a central type of nodes.

The text-attached heterogeneous information network will be the input from which we mine latent entity structures. The algorithms studied in this thesis do not always require the existence of both text and links, so they can work when certain information is missing.

The next section introduces the notion of latent entity structures.

1.3 Latent entity structure

We first introduce two important notions in order to define the latent entity structure.

Position: the location of a node or class of nodes in an information network formed of relationships. For example, every researcher has a unique location in the academic family forest formed by the advisor-advisee relationship. In general, multiple actors may share a common position.

Expectation: an evaluative standard applied to an incumbent of a position, such as rights, duties, norms and behavior, that a entity has to face and to fulfill. For example, the advisee is expected to publish papers with his/her advisor during the advising period.

The latent entity structure can then be defined as two parts: the positions and the expectations that are coupled to the positions. Typically, one is interested in finding out the positions of entities by evaluating their behavior data with the expectations. When the expectations are also unknown or partially unknown, one needs to find out the expectations as well from the data.

The key of entity structure discovery is finding the right position of an entity, whether expectations are known. Position has two aspects in specification, namely *relational* and *situational*. In a relational specification, the focal position is specified by its relationship to one or multiple counter positions. For example, a school superintendent has relationships with many other positions such as school board member, principal and teacher. Situational specification describes the scope of the information network in which the position is to be studied. For example, the superintendent position can be studied in a specific community, in the state of Illinois, or in the United States. It is necessary to specify at which level one intends to work.

1.3.1 Categorization

The fundamental element of entity structure mining is to determine the position of every single entity. However, in many cases, it is uninteresting, or infeasible to find the connection between all the entities and all the positinos. Depending on what is of larger interest, most mining tasks can be abstracted as one of the following questions.

A) Given a set of entities X, what are their positions?

B) Given a set of positions Y, which entities have these positions?

The first type of questions ask the position of one or multiple actors. For example, An investigator can specify several predefined positions and their expectations, and ask for the position labels of X, such as to label one's Facebook friends as relatives, schoolmates, colleagues or other friends. A mining system must answer the question for every actor in X. The second type of questions ask which actors have a certain role. The answers are often given in a comparative sense — some actors are more likely to have a position than others. A mining system should answer the question for every position in Y, but not necessary to cover all the actors.

Both types of questions can be studied in unsupervised and supervised settings, depending on whether the position of any entity is known.

1.3.2 Historical background

The concept of role has assumed a key position in the fields of sociology, social psychology, and cultural anthropology. Yet, despite its frequent use and presumed heuristic utility since 1920s and 1930s, the conceptualization of role is lack of progress until 1950s. The debate has been on whether role is a redundant concept in sociology since then. Even without a consensus of the exact definition of social roles, the operational problem of role analysis has attracted a lot of research interests. Recently, with the emergence of large scale social network data and analysis, computer scientists can contribute to the role discovery problem with new computing techniques.

We attempt to generalize the concept of role in sociology study and define the entity structure mining problem towards the interest and view of computer scientists. We keep the two most important and widely accepted concepts 'position' and 'expectation', but leave the other concepts that is not necessary for our computing technique. Also, we extend these definitions in heterogeneous information networks, not just homogeneous social networks. Not only persons but also other types of entities can hold positions. The expectation is not only about the homogeneous link structure, but also can be expressed by the heterogeneous linked entities and unstructured text.

1.4 Framework

This section presents a framework for mining hierarchical topics and revealing entity roles and relations. It is comprised of the following modules.

1.4.1 Hierarchical topic and community discovery

This is related to the situational specification of position. A position is only meaningful in certain scope of a large network. By mining topical communities from a network, one can use the communities as the situational specification of position. For example, in order to mine opinion leaders, one need to specify the scope because different communities may have different opinion leaders. The communities formed by research topics can provide suitable context for analyzing who the important contributors in each community are.

In particular, we explore the hierarchical structure of the topics. It exists in many domains we consider in this thesis. Also, the hierarchical structure is effective and efficient for both humans and machines to manipulate in many applications. Therefore, the function of this module is to construct a topical hierarchy from a text-attached heterogeneous information network. In the proposed method, each topic can be enriched with ranked lists of entities, which has two advantages: i) the entities provide additional informative context for each topic in the hierarchy and ii) the entity positions in the hierarchy are discovered and ready to query as we construct the hierarchy.

1.4.2 Topical phrase mining

To make sense of the mined topics we need to visualize them with human-interpretable phrases. For example, the topic of query processing and optimization may be described by the phrases {'query processing', 'query optimization',...}, while its parent topic of general problems in databases may be described by {'query processing', 'database systems', 'concurrency control',...}. This module automatically mine phrases from the text and rank them according to their representativeness of each topic.

1.4.3 Entity topical role analysis

After the topical community is discovered, we can analyze the roles of entities in a desired scope. There are two types of questions we aim to answer, corresponding to the Type-A and Type-B questions in Section 1.3.1.

- Type-A question: for a given topical community and a specific entity, what is that entity's role in the community? For instance, which topics within the community get published in a particular conference? Or, which specific topics within the community does an author contribute to?
- Type-B question: given a topical community and an entity type, which entities play the most important roles in the community? For example, an author's contribution to the topics of a community (by publishing papers) represents the author's role in that community.

1.4.4 Entity relationship mining

This is related to the relational specification of position. A focal position can be described through its relation with one or multiple counter positions. For example, the advisor and advisee can be two positions that manifest each other through the advising relation. If one can discover the relationship between any pair of entities, their position can then be determined.

In this thesis, we studied in depth a particular form of relationship in which entities form a tree structure along the relationship. It exists in quite a few domains, such as hyponymy (is-a) relationship, advisor-advisee and manager-subordinate.

In sum, the main features of the framework are:

• *Recursive hierarchy construction*: The hierarchy is constructed in a top-down order. One can recursively apply our method to expand the hierarchy. It is flexible to revise part of the hierarchy while remaining other parts intact.

• *Phrase-represented topic*: Each topic is represented by a ranked list of topical phrases, such that a child topic is a subset of its parent topic. For example, the topic of query processing and optimization may be described by the phrases {'query processing', 'query optimization',...}, while its parent topic of general problems in databases may be described by {'query processing', 'database systems', 'concurrency control',...}. The phrases are automatically mined from the text and ranked according to their representativeness of each topic.

• *Entity-embedded topic*: The linked entity information enhances the topic discovery solely based on textual information. For example, the authors and venues linked to each paper help finding its topics. In our framework, each topic can be enriched with ranked lists of entities, which has two advantages: i) the entities provide additional informative context for each topic in the hierarchy and ii) the entity positions in the hierarchy are discovered and ready to query as we construct the hierarchy.

• *Dependency modeling of entity relations*: Utilizing heterogeneous links and text information, our method is able to capture a variety of semantic signals, including constraints and dependencies, to recover the relationship with the best known accuracy. These signals are categorized according to semantics, and formulated with simple and unified mathematical forms.

The rest of the thesis is organized as follows. Chapter 2 reviews the literature. Then the above modules are presented in Chapter 3 to 6. Chapter 7 focuses on solving the computational challenge to scale up the method. Chapter 8 discusses the future work.

Chapter 2

Literature Review

2.1 Traditional topic modeling

Statistical topic modeling techniques model documents as mixtures of multiple topics, while every topic is modeled as a distribution over words. Two important models are PLSA (probabilistic latent semantic analysis) [42] and its Bayesian extension LDA (latent Dirichlet allocation) [14]. They model the generative processes of the words for all the documents in a corpus. To generate each word, a latent topic label is first chosen from a pool of 'flat topics' with a multinomial distribution. And then a word is sampled according to this topic's word distribution. With these generative assumptions, the unknown word distribution of every topic can be inferred so as to best explain the observed word occurrences in the documents.

Most of existing topic model inference methods are based on the *maximum likelihood* (ML) principle (including its Bayesian version *maximum a posterior*). For example, PLSA [42] uses an Expectation-Maximization algorithm to approximately optimize the data likelihood. For LDA, two most popular approximate inference methods have been variational Bayesian inference [14] and Markov Chain Monte Carlo (especially Gibbs sampling) [33]. In spite of the vast body of followup work, the computational complexity of ML inference is not studied until 2011. Sontag and Roy [76] show that the document-level inference is not always well defined, and Arora, Ge and Moitra [8] prove the NP-hardness of exact corpus-level ML inference.

In accordance with their theoretical hardness, the above inference methods tend to suffer from slow convergence and long runtime. As a result, there has been a substantial amount of work targeting on accelerating the above methods. *e.g.*, by leveraging sparsity [68, 98, 40] and parallelization [65, 74, 100], or online learning mechanism [3, 39, 29]. However, none of them have theoretical guarantee of convergence within a bounded number of iterations, and are nondeterministic either due to sampling or the random initialization.

Recently, an alternative inference of topic models has been proposed based on the *method of moments* [5], and improved in [6]. Compared with ML inference, it has the following two advantages: i) the distance between inferred corpus-level parameters and the true parameters has a theoretical upper bound that is inversely related to sample size; ii) the convergence is guaranteed with a bounded number of iterations. However, the practical issue of scalability has not been solved in the theoretical work. Another related study [8] assumes the existence of *anchor word* that only exists in one topic, and uses that assumption to bound the recovery error. Its efficiency is improved in [9]. This method requires stronger assumptions than [5] and the error bound is weaker.

2.2 Extension of topic models

Traditional topic modeling technique was proposed for flat topics in bag-of-words text. It is challenging to design a framework that has all the three advanced features: hierarchical topics, phrase-represented topics and entity-enriched topics, while it is relatively easier to develop a method that has partial features. This section reviews the methdologies that has added to traditional topic modeling each of these features. My thesis establishes the first framework that enables all these features.

2.2.1 Hierarchical topic models

Hierarchical topic models follow the same generative spirit. Instead of generating from a pool of flat topics, these models assume an internal hierarchical structure of the topics. Different models use different generative processes to simulate this hierarchical structure: nested Chinese Restaurant Process [32], Pachinko Allocation [53], Hierarchical Pachinko Allocation [61], recursive Chinese Restaurant Process [48], and nested Chinese Resturant Franchise [4]. When these models are applied to constructing a topical hierarchy, the entire hierarchy must be inferred all at once from the corpus.

Few of the speedup ideas introduced in Section 2.1 have been adopted by the hierarchical topic model studies.

All of the hierarchical topic models follow the bag-of-words assumption.

There are a few alternative approaches to constructing a topical hierarchy. Pujara and Skomoroch [70] proposed to first run LDA on the entire corpus, and then split the corpus heuristically according to the results and run LDA on each split corpus individually. It is a recursive approach without an integrated generative process, so the robust recovery property is not applicable. Recursive clustering is used to cluster documents [31], queries [55], keywords [90] *etc.*, to construct hierarchies of different kinds.

2.2.2 Phrase-enhanced topic models

Recently many attempts have been made to relax the 'bag-of-words' assumption of traditional topic models. These topical phrase extraction techniques fall into three main categories: i) those that infer phrases and topics simultaneously by creating complex generative models; ii) those that apply topical phrase discovery as a post-process to topic model inference; and iii) those that apply phrase mining as a pre-process of topic modeling.

We first review the methods in the first category. [82] experimented with incorporating a bigram language model into LDA. TNG [93] is a state-of-the-art approach to n-gram topic modeling that uses additional latent variables and word-specific multinomials to model bi-grams. These bigrams can be combined to form n-gram phrases. PD-LDA [54] uses a hierarchal Pitman-Yor process to share the same topic among all words in a given n-gram. Because PD-LDA uses a nonparametric prior to share a topic across the words in an n-gram, it can be considered a natural generalization of the LDA bigram language model to n-grams.

It is obvious that the scalability and robustness issues of bag-of-words topic models will become more challenging in these integrated models. A practical approach is to separate the topic modeling part and the phrase mining part. In the second category, Turbo Topics [12] uses a back-off n-gram model and permutation tests to assess the significance of a phrase. The tests are time consuming. Zhao et al. [101] uses a graph-based ranking technique to rank the importance of unigrams and then heuristically combine the score of the constituents of phrases for ranking them. In the third category, [46] attempts to directly using frequent patterns to enrich the text corpora for topic modeling. The objective of this method is to enrich the overall quality of the topic model and not for the creation of interpretable topical phrases.

2.2.3 Entity-enriched topic models

Recently, researchers have studied how to mine topics when documents have additional links to multiple typed entities. These approaches make use of multi-typed entities in three different ways: i) resemble entities to documents; ii) resemble entities to topics; and iii) resemble entities to words.

In the first category of work, the entities are treated like documents – each entity has a multinomial distribution over topics. iTopicModel [77] and TMBP–Regu [25] use links to regularize the topic distribution so that linked documents or entities have similar topic distributions. Chen et al. [18] extends LDA to use entities as additional sources of topic choices for each document. Tang et al. [79] argue that this kind of extension has a problem of ‘competition for words’ among different sources when the text is sparse. They propose to aggregate documents linked to a common entity as a pseudo document, and regularize the topic distributions inferred from different aggregation views to reach a consensus.

In the second category of work, the entities are treated like topics – each entity has a multinomial distribution over words. Kim et al. [47] assumes each word is generated from a distribution that is decided by a pair of (topic, entity). And the distribution corresponding to that pair is generalized from a Dirichlet prior which is the linear combination of the distribution for the topic and the distribution for the entity.

In the third cateogory of work, the entities are treated like words – each topic has a multinomial distribution over entities. Entities are additional elements te be generated for each document. Most models, including Conditionally independent LDA [41], SwitchLDA [64] and NetClus [78], assumes conditional independence of words and entities given topics. Two models, CorrLDA1 [13] and CorrLDA2 [64] break that assumption.

The first category of work uses entities to regularize textual topic discovery, but the goal is still to infer meaningful topics represented by word distributions. The third cateogry of work can further enrich the topic representation with entities. The second category of work has the ability to infer a profile for each entity with high degree of freedom, but the number of parameters is orders of magnitude larger.

2.3 Mining relations

In a broad view of relationship identification, there are studies in different domains. One category of such work is relation mining from text data. Text mining and language processing techniques have been employed on text data to add structured semantic information to plain text. Commonly referred entity relationship extraction belongs to this category, such as those developed around the Knowledge Base Population task [44]. While most of them explore low-level textual features [1, 34], some also exploit relational patterns for reasoning [71, 22]. Another line of studies focus primarily on processing the interaction events in a social network, and try to discover relationships from traffic or content patterns of the interaction, *e.g.*, from the email communication data [58, 27]. The central problem for most of these studies is judging whether a pair of objects have a certain relationship. They do not have special requirement for discovered relationship to form certain structures.

One particular relationship considered in this thesis is asymmetric among a set of linked objects, and the objects can be organized in a tree-like structure along this relationship. In a few recent studies, finding such a relationship is an essential task. Leskovec *et al.* [50] defines the *DAG partitioning problem*, which is NP-hard, and proposes a class of heuristic methods of finding a parent for each non-rooted node, which can be regarded as finding hierarchical relationships among the phrase quotations in news articles. Kemp and Tenenbaum [45] propose to use a generative model to find structure in data, and Maiya and Berger-Wolf [57] apply the similar idea in inferring the social network hierarchy with the maximum likelihood of observing the interactions among the people. In the Web search domain, Yin and Shah [99] study the problem of building taxonomies of search intents for entity queries based on inferring the "belonging" relationships between them with unsupervised approaches. There is other related work for building taxonomies from Web tags with similar methodology [26]. NLP Researchers have also studied entity hyponymy (or is-a) relation from web documents. Wong et al. [95] provides a comprehensive survey. These approaches rely on one kind of observation data, like links or text, and a clear standard of tree construction. This thesis considers heterogeneous data with text and multiple types of links while no single factor determines the hierarchy.

In Information Retrieval community, researchers developed supervised methods to discover the "reply" relationship for online conversations, which is also a hierarchical relationship [73]. They use domain-specific features and could not be directly applied in more general scenario. My work made the first attempt to formalize and solve the general hierarchical relationship learning problem.

Chapter 3

Hierarchical Topic and Community Discovery

Automated organization of the topics from unstructured and interconnected data at different levels of granularity is an important problem in knowledge engineering with many valuable applications such as information summarization, search and online analytical processing (OLAP). With vast amount of data and dynamic change of users' need, it is too costly to rely on human experts to do manual annotation and provide ready-to-use topical hierarchies. Thus it is critical to create a robust method for automated construction of high-quality topical hierarchies from text-rich heterogeneous information networks. It is one of the key tasks in the framework of mining latent entity structures.

The proposed method discovers the hierarchical topics recursively. That is, to grow the tree gradually from the root. A key operation of the methodology is to mine subtopics of a topic represented by a leaf vertex in the current tree, so that vertices that represent these subtopics will be added as children of this vertex. Repeating this operation will grow the tree recursively in a top-down manner. The tree constructed at any time during this process encodes the discovered topics so far, and can be used as output of the mined hierarchical topics.

Compared with previous non recursive hierarchical topic modeling techniques, the recursive strategy has the following advantage.

- The parent-child relation is ensured during the recursion. In non-recursive methods, the parent-child relation may not be obvious due to the sampling along long paths.
- One can stop expanding the hierarchy after an arbitrary number of growing operations as needed. In non-recursive methods, all topics must be inferred before the algorithm stops.
- It is flexible to revise part of the hierarchy while remaining other parts intact. In non-recursive methods, the change of the shape of the hierarchy may result in entirely different hierarchy.

- As a network-based analysis method, it can be extended naturally to heterogeneous networks, with or without text. The non-recursive methods have intricated generative processes and are hard to extend.

For the data where no text information is available, our method can be applied to find hierarchical community structures.[1]

3.1 Generative model for text or homogeneous network

In this section we propose a method for recursively discovering hierarchical topics from pure text. The method is based on analysis with term co-occurrence networks, which are converted from the collection of documents. The analysis technique can be applied to generic homogeneous network (with only one type of nodes and links).

Definition 2 (Topical hierarchy) *A topical hierarchy is defined as a tree $\mathcal{T}$ in which each node is a topic. Every non-leaf topic t has C_t child topics. Topic t is characterized by a probability distribution over words ϕ_t, and visualized as an ordered list of phrases $\mathcal{P}_t = \{P_{t,1}, P_{t,2}, \dots\}$, where $P_{t,i}$ is the phrase ranked at i-th position for topic t.*

The number of child topics C_t of each topic can be specified as input, or decided automatically by the construction method. In both cases, we assume it is bounded by a small number K, such as 10. This is for efficient browsing of the topics. The number K is named the *width* of the tree $\mathcal{T}$.

For convenience, we denote a topic using the top-down path from root to this topic. The root topic is denoted as o. Every non-root topic t is denoted by π_t/χ_t, where π_t is the notation of its parent topic, and χ_t is the index of t among its siblings. For example, the topic $t1$ in Figure 3.1 is denoted as $o/1$, and its child $t5$ is denoted as $o/1/2$. The *level* h_t of a topic t is defined to be the number of '/' in its notation. So root topic is in level 0, and $t5$ is in level 2. The *height* h of a tree is defined to be the maximal level over all the topics in the tree. Clearly, the total number T of topics is upper bounded by $\frac{K^{h+1}-1}{K-1}$.

Now we present our network analysis technique for topical hierarchy construction. Every topic node t in the topical hierarchy is associated with a term co-occurrence network G^t. The root node o

[1]The content of this chapter is largely based on Wang et al. [88] and Wang et al. [89].

is associated with the term co-occurrence network G^o constructed from the collection of documents. G^o consists of V nodes and a set of links $\mathcal{E}$. A node $v \in [V]$ represents a term, and a link (i, j) between two nodes represents a co-occurrence of the two terms in a document. The number of links $e_{ij} \in \mathcal{E}$ between two nodes i and j is equal to the number of co-occurrences of the two terms. For every non-root node $t \neq o$, we construct a subnetwork G^t by clustering the term co-occurrence network of its parent π_t. G^t has all of the nodes from G^{π_t}, but only those links belonging to the particular subtopic t.

We chose to use term co-occurrence network for topic analysis instead of document-term topic modeling because the it naturally supports recursive mining: the clustering result for one topic can be used as the input when further partitioning the topic into subtopics. We name this method CATHY (construct a topical hierarchy). It generates a topical hierarchy in a top-down, recursive way:

Step 1. Construct the term co-occurrence network G^o from the document collection. Set $t = o$.

Step 2. For a topic t, cluster the term co-occurrence network G^t into subtopic subnetworks $G^{t/z}, z \in [C_t]$.

Step 3. Recursively apply Steps 2 to each subtopic $t/z, z \in [C_t]$ to construct the hierarchy in a top-down fashion.

In the following, we introduce the process of clustering for one topic t. We assume $C_t = k$. The value of k can be either specified by users or chosen using a model selection criterion such as the Bayesian Information Criterion [72].

In the term co-occurrence network G^t, we assume every co-occurrence of two terms i and j is attributed to a topic $t/z, z \in [C_t] = [k]$. Thus, the total link frequency e_{ij} between i and j is a summation of the number of links between i and j in each of the k topics: $e^t_{ij} = \sum_{z=1}^{k} e^{t/z}_{ij}$. The goal is thus to estimate $e^{t/z}_{ij}$ for $z = 1 \dots k$, which is unlike most network analysis approaches.

To generate a topic-t/z link, we first generate one end node i following a multinomial distribution $p(i|t/z) = \phi_{t/z,i}$, and then generate the other end node j with the same multinomial distribution $p(j|t/z) = \phi_{t/z,j}$. The probability of generating a topic-z link (i, j) is therefore $p(i|t/z)p(j|t/z) = \phi_{t/z,i}\phi_{t/z,j}$.

With this generative assumption for each individual link, we can derive the distribution of topical frequency for any two terms (i, j). If we repeat the generation of topic-z links for $\rho_{t,z}$ iterations, then the chance of generating a particular topic-t/z link between i and j can be modeled as a Bernoulli trial with success probability $\phi_{t/z,i}\phi_{t/z,j}$. When $\rho_{t,z}$ is large, the total number of successes $e_{ij}^{t/z}$ approximately follows a Poisson distribution $Pois(\rho_{t,z}\phi_{t/z,i}\phi_{t/z,j})$.

Now we can write down the generative model for random variables $e_{ij}^{t/z}$ with parameters $\rho_{t,z}, \phi_{t/z}$.

$$e_{ij}^{t/z} \sim Poisson(\rho_{t,z}\phi_{t/z,i}\phi_{t/z,j}), z = 1, \ldots, k \tag{3.1}$$

$$\sum_{i=1}^{V} \phi_{t/z,i} = 1, \phi_{t/z,i} \geq 0, \rho_{t,z} \geq 0 \tag{3.2}$$

The constraints guarantee a probabilistic interpretation. According to the *expectation* property of the Poisson distribution, $E(e_{ij}^{t/z}) = \rho_{t,z}\phi_{t/z,i}\phi_{t/z,j}$. Also, according to the *additive* property of expectations,

$$E(\sum_{i,j} e_{ij}^{t/z}) = \sum_{i,j} \rho_{t,z}\phi_{t/z,i}\phi_{t/z,j} = \rho_{t,z} \sum_{i} \phi_{t/z,i} \sum_{j} \phi_{t/z,j} = \rho_{t,z}$$

In other words, $\rho_{t,z}$ is the total expected number of links in topic t/z.

One important implication due to the *additive* property of Poisson distribution is that

$$e_{ij}^{t} = \sum_{z=1}^{k} e_{ij}^{t/z} \sim Poisson(\sum_{z=1}^{k} \rho_{t,z}\phi_{t/z,i}\phi_{t/z,j}) \tag{3.3}$$

So given the model parameters, the probability of all observed links is

$$\begin{aligned} p(\{e_{ij}^{t}\}|\phi, \rho) &= \prod_{i,j\in[V]} p(e_{ij}^{t}|\phi_i, \phi_j, \rho) \\ &= \prod_{i,j\in[V]} \frac{(\sum_{z=1}^{k} \rho_{t,z}\phi_{t/z,i}\phi_{t/z,j})^{e_{ij}^{t}} \exp(-\sum_{z=1}^{k} \rho_{t,z}\phi_{t/z,i}\phi_{t/z,j})}{e_{ij}^{t}!} \end{aligned} \tag{3.4}$$

In this model, the observed information is the total number of links between every pair of nodes, including zero links and self-links. The parameters which must be learned are the role of each node in each topic $\phi_{t/z,i}, i \in [V], z = 1, \ldots, k$, and the expected number of links in each topic $\rho_{t,z}$. The total number of free parameters to learn is therefore kV. We learn the parameters by the *Maximum Likelihood* (ML) principle: find the parameter values that maximize the likelihood in Eq. (3.4). We use an Expectation-Maximization (EM) algorithm that can iteratively infer the model parameters:

$$\text{E}-\text{step}: \quad \hat{e}_{ij}^{t/z} = e_{ij}^{t} \frac{\rho_{t,z}\phi_{t/z,i}\phi_{t/z,j}}{\sum_{x=1}^{k}\rho_{t,x}\phi_{t/x,i}\phi_{t/x,j}} \tag{3.5}$$

$$\text{M}-\text{step}:$$

$$\rho_{t,z} = \sum_{i,j} \hat{e}_{ij}^{t/z} \tag{3.6}$$

$$\phi_{t/z,i} = \frac{\sum_j \hat{e}_{ij}^{t/z}}{\rho_{t,z}} \tag{3.7}$$

Intuitively, the E-step calculates the expected number of links $\hat{e}_{ij}^{t/z}$ in each topic t/z between the terms i and j: the ratio of $\hat{e}_{ij}^{t/z}$ to e_{ij}^{t} is proportional to its Poisson parameter $\rho_{t,z}\phi_{t/z,i}\phi_{t/z,j}$. The M-step calculates the ML parameter estimates: $\phi_{t/z,i}$ is the ratio of the total number of links in topic t/z where one end node is i and $\rho_{t,z}$, which is the sum of the total expected number of links in topic t/z.

We update $\hat{e}_{ij}^{t/z}, \phi_{t/z,i}, \rho_{t,z}$ in each iteration. Note that if $e_{ij}^{t} \notin \mathcal{E}$, we do not need to calculate $\hat{e}_{ij}^{t/z}$ because it equals 0. Therefore, the time complexity for each iteration is $\mathcal{O}((|\mathcal{E}| + V)k) = \mathcal{O}(|\mathcal{E}|k)$. Like other EM algorithms, the solution converges to a local maximum and the result may vary with different initializations. The EM algorithm may be run multiple times with random initializations to find the solution with the best likelihood. We empirically find that the EM algorithm generally requires hundreds of iterations to converge. A more scalable method will be introduced in Chapter 7.

It is important to note that our method naturally supports top-down hierarchical clustering. To further discover subtopics of a topic, we can extract the subnetwork where $\mathcal{E}^{t/z} = \{\hat{e}_{ij}^{t/z} | \hat{e}_{ij}^{t/z} \geq 1\}$ (expected number of links attributed to that topic, ignoring values less than 1) and then apply the same generative model on the subnetwork. This process can be recursively repeated until the desired hierarchy is constructed.

After the hierarchy is discovered, we can use the technique in next chapter to visualize the topics with topical phrases. A sample output with the DBLP publication titles is visualized in

Figure 3.3.

3.2 Generative model for heterogeneous network

A variety of existing work is devoted to constructing topical hierarchies from text data. However, few approaches utilize link information from typed entities that may be present in the data. Conversely, existing methods for heterogeneous network analysis and topic modeling have demonstrated that multiple types of linked entities improve the quality of topic discovery (*e.g.*, NetClus [78]), but these methods are not designed for finding hierarchical structures (See Figure 3.2 for an example output of NetClus). Therefore, there is no existing method that is able to construct a multi-typed topical hierarchy from a text-attached heterogeneous network.

In this section, we develop a method that makes use of both textual information and heterogeneous linked entities to automatically construct multi-typed topical hierarchies. It is an extension of the generative model for text in the last section.

Formally, every topic node t in the topical hierarchy is associated with an edge-weighted network $G^t = (\{\mathcal{V}_x^t\}, \{\mathcal{E}_{x,y}^t\})$, where $\mathcal{V}_x^t$ is the set of type-x nodes in topic t, and $\mathcal{E}_{x,y}^t$ is the set of link weights between type x and type y nodes (x and y may be identical) in topic t. $e_{i,j}^{x,y,t} \in \mathcal{E}_{x,y}^t$ represents the weight of the link between node v_i^x of type x and node v_j^y of type y.

To construct the network G^o for the root topic o, we convert the text-attached heterogeneous information network $H = (\{\mathcal{V}_x\}, \{\mathcal{E}_{x,y}\}, \mathcal{D})$ into a collapsed network. Assume there are m node types and m^2 link types in H. Type-1 nodes are documents. G^o is obtained by converting the documents into term co-occurrence links and adding them into the network in H. $G^o = (\{\mathcal{V}_x\}_{x=2}^m \cup \{\mathcal{V}_{m+1} = [V]\}, \{\mathcal{E}_{x,y}\}_{x,y=2}^m \cup \{\mathcal{E}_{x,m+1}\}_{x=2}^{m+1})$. The $(m+1)$-th type of nodes in G^o are terms. If an entity is linked to a document in H, then it is linked with all the words in that document. The link weight between an entity and a word in G^o is equal to the summation of the link weight between the entity and all the documents in H.

Example 3.1 *We can construct a collapsed network from the research publications, with $m = 3$ types of nodes: term, author and venue; and $l = 5$ types of links: term-term, term-author, term-venue, author-author, author-venue. The link weight between every two nodes is equal to the number* *of papers where the two objects co-occur.*

When there is only text information, the collapsed network reduces to term co-occurrence network, as we discussed in the last section. When there is no text information, $G^o = H$.

For every non-root node $t \neq o$, we construct a subnetwork G^t by clustering the network G^{π_t} of its parent π_t. G^t inherits the nodes from G^{π_t}, but contains only the fraction of the original link weights that belongs to the particular subtopic t. Figure 3.5 visualizes the weight of each link in each network and subnetwork by line thickness (disconnected nodes and links with weight 0 are omitted).

This method is named CATHYHIN (construct a topical hierarchy from heterogeneous information network). It generates a heterogeneous topical hierarchy in a top-down, recursive way:

Step 1. Construct the edge-weighted network G^o. Set $t = o$.

Step 2. For a topic t, cluster the network G^t into subtopic subnetworks $G^{t/z}, z \in [C_t]$ using a generative model.

Step 4. Recursively apply Steps 2 to each subtopic $t/z, z \in C_t$ to construct the hierarchy in a top-down fashion.

After the hierarchy is discovered, we can use the technique in next chapter to visualize the topics with topical phrases, and analyze entity roles using the technique in Chapter 5. A sample output for the DBLP network is visualized in Figure 3.4.

In Section 3.2.1, we describe our unified generative model and present an inference algorithm with a convergence guarantee. In Section 3.2.2, we further extend our approach to allow different link types to play different degrees of importance in the model (allowing the model to, for example, decide to rely more on term co-occurrence information than on co-author links).

3.2.1 The basic model

We first introduce the basic generative model, which considers all link types to be equally important. For a given topic t, we assume $C_t = k$. The value of k can be either specified by users or chosen using a model selection criterion. We discuss the choice of k in Section 3.2.3.

We assume every link has a direction. For undirected networks, we convert them to directed networks by duplicating each link between v_i^x and v_j^y in both directions $v_i^x \to v_j^y$ and $v_j^y \to v_i^x$. So our model can be applied to both undirected and directed networks.

Similar to NetClus, we assume each node type x has a multinomial distribution $\phi^{x,t/z}$ in each subtopic $t/z, z \in [k]$, such that $\phi^x_{t/z,i}$ is the importance of node v^x_i in topic t/z, subject to $\sum_i \phi^x_{t/z,i} = 1$. Each node type x also has a multinomial distribution $\phi^x_{t/0}$ for the background topic. In contrast to NetClus, we softly partition the link weights in G^t into subtopics. We model the generation of links so that we can simultaneously infer the partition of link weights (clustering) and the node distribution (ranking) for each topic.

Example 3.2 *In a computer science publication network, each of the three node types term, author and venue has a ranking distribution in each topic in the hierarchy. The distributions for a hypothetical topic about database may be: i) term - {database: 0.01, system: 0.005, query: 0.004, …}; ii) author - {Sujarit Chaudhuri: 0.03, Jeffery F. Naughton: 0.02, …} and iii) venue - {SIGMOD: 0.2, VLDB: 0.25, …}*

To derive our model, we first assume the links between any two nodes can be decomposed into one or multiple unit-weight links (*e.g.*, a link with weight 2 can be seen as a summation of two unit-weight links). Later we will discuss the case where the link weight is not an integer. Each unit-weight link has a topic label, which is either a subtopic $t/z, z \in [k]$, or a dummy label $t/0$, implying the link is generated by a background topic and should not be attributed to any subtopic of t.

The generative process for a link with unit weight is as follows:

1. Generate the topic label z according to a multinomial distribution ρ_t.
2. Generate the link type (x, y) according to a multinomial distribution θ.
3. If $z \in [k]$,
   (a) Generate the first end node u_1 from the type-x ranking distribution $\phi^x_{t/z}$.
   (b) ratene,e the second end node u_2 from the type-y ranking distribution $\phi^y_{t/z}$.

   Else $(z = 0)$

   (a) Generate the first end node u_1 from the type-x ranking distribution $\phi^x_{t/0}$.

(b) Generate the second end node u_2 from the type-y ranking distribution ϕ_t^y.

Example 3.3 *A link between two terms* query *and* processing *in a topic $t/z = t/1(Database)$ may be generated in the following order: i) generate the topic label $z = 1$ with probability $\rho_{t,z} = 0.2$; i) generate the link type $(x, y) = (term, term)$ according to $\theta_{x,y} = 0.15$; iii) generate the first end node $u_1 = query$ from the term distribution $\phi^x_{t/z,u_1} = 0.004$; and iv) generate the second end node $u_2 = processing$ from the term distribution $\phi^y_{t/z,u_2} = 0.001$.*

This process is repeated for M^t times, to generate all the unit-weight links. Note that when generating a background topic link, the two nodes i and j are not symmetric. The first end node is a background node, and can have a background topic link with any other nodes based simply on their importance in the parent topic, irrespective of any subtopic. Highly ranked nodes in the background topic tend to have a link distribution over all nodes that is similar to their distribution in the parent topic. This part can be altered if the network follows a different assumption about the background nodes. See Figure 3.6 for a graphical representation of the model.

With these generative assumptions for each unit-weight link, we can derive the distribution of link weight for any two nodes (v_i^x, v_j^y). First, we notice that the total number of topic-t/z unit-weight links is expected to be $M^t\rho_{t,z}$. It implies that we are expected to repeat the generation of topic-t/z unit-weight links for $M^t\rho_{t,z}$ times. Second, we investigate how many topic-t/z unit-weight links will be generated between two certain nodes v_i^x and v_j^y, among all the topic-t/z links. Consider the event 'the two end nodes are v_i^x and v_j^y' for each generated topic-t/z link. It is a Bernoulli trial with success probability $\theta_{x,y}\phi^x_{t/z,i}\phi^y_{t/z,j}$ for $z \in [k]$. When $M^t\rho_{t,z}$ is large, the total number of successes $e_{i,j}^{x,y,t/z}$ asymptotically follows a Poisson distribution $Pois\left(M^t\rho_{t,z}\theta_{x,y}\phi^x_{t/z,i}\phi^y_{t/z,j}\right)$. Similarly, the total number of background topic links $e_{i,j}^{x,y,t/0}$ asymptotically follows a Poisson distribution $Pois\left(M^t\rho_{t,0}\theta_{x,y}\phi^x_{t/0,i}\phi^y_{t,j}\right)$.

One important implication due to the *additive* property of Poisson distribution is:

$$e_{i,j}^{x,y,t} = \sum_{z=0}^{k} e_{i,j}^{x,y,t/z} \sim Poisson\left(M^t\theta_{x,y}s_{i,j}^{x,y,t}\right) \tag{3.8}$$

where $s_{i,j}^{x,y,t} = \sum_{z=1}^{k}\rho_{t,z}\phi^x_{t/z,i}\phi^y_{t/z,j} + \rho_{t,0}\phi^x_{t/0,i}\phi^y_{t,j}$.

This leads to a 'collapsed' model as depicted in Figure 3.7. Though we have so far assumed the link weight to be an integer, this collapsed model remains valid with non-integer link weights (due to Lemma 3.1, discussed in Section 3.2.2).

Given the model parameters, the probability of all observed links is:

$$p\left(\{e_{i,j}^{x,y,t}\}|\theta,\rho,\phi\right) = \prod_{v_i^x, v_j^y} \frac{\left(M^t\theta_{x,y}s_{i,j}^{x,y,t}\right)^{e_{i,j}^{x,y,t}} \exp\left(-M^t\theta_{x,y}s_{i,j}^{x,y,t}\right)}{e_{i,j}^{x,y,t}!} \tag{3.9}$$

We learn the parameters by the *Maximum Likelihood* (ML) principle: find the parameter values that maximize the likelihood in Eq. (3.9). First, we take the logarithm and remove the part independent of the parameters:

$$\begin{aligned} L(\theta,\rho,\phi) &= \sum_{v_i^x,v_j^y} \left[e_{i,j}^{x,y,t}\log\left(\theta_{x,y}s_{i,j}^{x,y,t}\right) - M^t\theta_{x,y}s_{i,j}^{x,y,t}\right] \\ &= \sum_{v_i^x,v_j^y} e_{i,j}^{x,y,t}\left[\log\theta_{x,y} + \log\left(\sum_{z=1}^k \rho_{t,z}\phi_{t/z,i}^x\phi_{t/z,j}^y + \rho_{t,0}\phi_{t/0,i}^x\phi_{t,j}^y\right)\right] \\ &- M^t\sum_{v_i^x,v_j^y}\theta_{x,y}\left(\sum_{z=1}^k \rho_{t,z}\phi_{t/z,i}^x\phi_{t/z,j}^y + \rho_{t,0}\phi_{t/0,i}^x\phi_{t,j}^y\right) \end{aligned}$$

The gradient method is hard to apply due to the logarithm of summation. We introduce an auxiliary distribution $q_{i,j}^{x,y}$ over subtopics for every link (v_i^x, v_j^y). They satisfy

$$\sum_{z=0}^k q_{i,j}^{x,y,z} = 1, \forall v_i^x, v_j^y \tag{3.10}$$

Due to Jensen's inequality, we have:

$$\begin{aligned} &\log\left(\sum_{z=1}^k \rho_{t,z}\phi_{t/z,i}^x\phi_{t/z,j}^y + \rho_{t,0}\phi_{t/0,i}^x\phi_{t,j}^y\right) \\ =&\log\left(\sum_{z=1}^k q_{i,j}^{x,y,z}\frac{\rho_{t,z}\phi_{t/z,i}^x\phi_{t/z,j}^y}{q_{i,j}^{x,y,z}} + q_{i,j}^{x,y,0}\frac{\rho_{t,0}\phi_{t/0,i}^x\phi_{t,j}^y}{q_{i,j}^{x,y,0}}\right) \\ \geq&\sum_{z=1}^k q_{i,j}^{x,y,z}\log\frac{\rho_{t,z}\phi_{t/z,i}^x\phi_{t/z,j}^y}{q_{i,j}^{x,y,z}} + q_{i,j}^{x,y,0}\log\frac{\rho_{t,0}\phi_{t/0,i}^x\phi_{t,j}^y}{q_{i,j}^{x,y,0}} \end{aligned}$$

where the equality holds if and only if:

$$\frac{\rho_{t,z}\phi^x_{t/z,i}\phi^y_{t/z,j}}{q^{x,y,z}_{i,j}} = \frac{\rho_{t,0}\phi^x_{t/0,i}\phi^y_{t,j}}{q^{x,y,0}_{i,j}}, \forall z = 1, \ldots, k \tag{3.11}$$

We define an auxiliary function $F(q, \theta, \rho, \phi)$:

$$\begin{aligned} F(q, \theta, \rho, \phi) = &\sum_{v^x_i, v^y_j} e^{x,y,t}_{i,j} \left(\log \theta_{x,y} + \sum_{z=1}^{k} q^{x,y,z}_{i,j} \log \frac{\rho_{t,z}\phi^x_{t/z,i}\phi^y_{t/z,j}}{q^{x,y,z}_{i,j}} + q^{x,y,0}_{i,j} \log \frac{\rho_{t,0}\phi^x_{t/0,i}\phi^y_{t,j}}{q^{x,y,0}_{i,j}} \right) \\ &- M^t \sum_{v^x_i, v^y_j} \theta_{x,y} \left(\sum_{z=1}^{k} \rho_{t,z}\phi^x_{t/z,i}\phi^y_{t/z,j} + \rho_{t,0}\phi^x_{t/0,i}\phi^y_{t,j} \right) \end{aligned}$$

F can be maximized by iteratively applying two alternating steps: i) Fix θ, ρ, ϕ, choose q to optimize F; ii) Fix q, choose θ, ρ, ϕ to optimize F. For i), Eq. (3.11) and Eq. (3.10) together imply:

$$q^{x,y,z}_{i,j}(\theta, \rho, \phi) = \frac{\rho_{t,z}\phi^x_{t/z,i}\phi^y_{t/z,j}}{\sum_{c=1}^{k} \rho_{t,c}\phi^x_{t/c,i}\phi^y_{t/c,j} + \rho_{t,0}\phi^x_{t/0,i}\phi^y_{t,j}}, z = 1, \ldots, k \tag{3.12}$$

$$q^{x,y,0}_{i,j}(\theta, \rho, \phi) = \frac{\rho_{t,0}\phi^x_{t/0,i}\phi^y_{t,j}}{\sum_{c=1}^{k} \rho_{t,c}\phi^x_{t/c,i}\phi^y_{t/c,j} + \rho_{t,0}\phi^x_{t/0,i}\phi^y_{t,j}} \tag{3.13}$$

For ii), we use the Lagrange multiplier method to incorporate the probability constraints for θ, ρ and ϕ. The gradient method yields a closed-form solution:

$$\theta_{x,y}(q) = \frac{\sum_{v^x_i, v^y_j} e^{x,y,t}_{i,j}}{M^t} \tag{3.14}$$

$$\rho_{t,z}(q) = \frac{\sum_{v^x_i, v^y_j} e^{x,y,t}_{i,j} q^{x,y,z}_{i,j}}{M^t} \tag{3.15}$$

$$\phi^x_{t/z,i}(q) = \frac{\sum_{v^y_j} (e^{x,y,t}_{i,j} q^{x,y,z}_{i,j} + e^{y,x,t}_{j,i} q^{y,x,z}_{j,i})}{\sum_{v^x_u, v^y_j} (e^{x,y,t}_{u,j} q^{x,y,z}_{u,j} + e^{y,x,t}_{j,u} q^{y,x,z}_{j,u})}, z = 1, \ldots, k \tag{3.16}$$

$$\phi^x_{t/0,i}(q) = \frac{\sum_{v^y_j} e^{x,y,t}_{i,j} q^{x,y,0}_{i,j}}{\sum_{v^x_u, v^y_j} e^{x,y,t}_{u,j} q^{x,y,0}_{u,j}} \tag{3.17}$$

During the iterations, the value of function F keeps non-decreasing. Let $q^{(a)}, \theta^{(a)}, \rho^{(a)}, \phi^{(a)}$

denote the parameter values after the a-th iteration. We then have:

$$L(\theta^{(a)}, \rho^{(a)}, \phi^{(a)}) = F(q^{(a+1)}, \theta^{(a)}, \rho^{(a)}, \phi^{(a)})$$

Therefore, the value of $L(\theta^{(a)}, \rho^{(a)}, \phi^{(a)})$ also keeps non-decreasing during the iterations. Since the function L is upper bounded, $L(\theta^{(a)}, \rho^{(a)}, \phi^{(a)})$ eventually converges to a local maximum.

This solution is similar to an Expectation-Maximization (EM) algorithm that is used for ML inference for many statistical models. In fact, we have the following theorem.

Theorem 3.1 *The solution Eq. (3.12)-(3.17) derived from the collapsed model (Figure 3.6) is equivalent to an EM solution derived from the unrolled model (Figure 3.7)*

Proof: In the unrolled model, the likelihood of a unit-weight link can be written as:

$$p(x, y, u_1, u_2|\theta, \rho, \phi, t) = p(x, y|\theta) \sum_z p(z|\rho_t) p(u_1|x, t/z, \phi) p(u_2|y, t/z, \phi)$$

in which the topic z is a latent variable. The EM algorithm iteratively applies the following two steps.

Expectation step (E-step). Calculate the expected value of the log likelihood function with respect to the conditional distribution of latent variables given observed variables under the current estimate of the parameters $\theta^{(a)}, \rho^{(a)}, \phi^{(a)}$.

$$\begin{aligned}
& Q(\theta, \rho, \phi|\theta^{(a)}, \rho^{(a)}, \phi^{(a)}) \\
= & \sum_{x,y,u_1,u_2} \sum_z p(z|x, y, t, u_1, u_2, \theta^{(a)}, \rho^{(a)}, \phi^{(a)}) \log p(x, y, u_1, u_2|t/z, \theta, \rho, \phi) \\
= & \sum_{x,y,u_1,u_2} \sum_z p(z|x, y, t, u_1, u_2, \theta^{(a)}, \rho^{(a)}, \phi^{(a)}) \log p(x, y|\theta) p(u_1|t/z, \phi) p(u_2|t/z, \phi)
\end{aligned}$$

Applying Bayes' theorem, we have:

$$\begin{aligned}
p(z|x,y,u_1,u_2,\theta^{(a)},\rho^{(a)},\phi^{(a)},t) &= \frac{p(z,x,y,u_1,u_2|\theta^{(a)},\rho^{(a)},\phi^{(a)},t)}{p(x,y,u_1,u_2|\theta^{(a)},\rho^{(a)},\phi^{(a)},t)} \\
=&\frac{p(x,y|\theta^{(a)}p(z|\rho_t^{(a)})p(u_1|t/z,\phi^{(a)})p(u_2|t/z,\phi^{(a)})}{\sum_c p(x,y|\theta^{(a)})p(c|\rho_t^{(a)})p(u_1|t/c,\phi^{(a)})p(u_2|t/c,\phi^{(a)})} \\
=&\frac{p(z|\rho_t^{(a)})p(u_1|t/z,\phi^{(a)})p(u_2|t/z,\phi^{(a)})}{\sum_c p(c|\rho_t^{(a)})p(u_1|t/c,\phi^{(a)})p(u_2|t/c,\phi^{(a)})} \\
=&\begin{cases} \frac{\rho_{t,z}\phi^x_{t/z,u_1}\phi^y_{t/z,u_2}}{\sum_{c=1}^k \rho_{t,c}\phi^x_{t/c,u_1}\phi^y_{t/c,u_2}+\rho_{t,0}\phi^x_{t/0,u_1}\phi^y_{t,u_2}} & z\in[k] \\ \frac{\rho_{t,0}\phi^x_{t/0,u_1}\phi^y_{t,u_2}}{\sum_{c=1}^k \rho_{t,c}\phi^x_{t/c,u_1}\phi^y_{t/c,u_2}+\rho_{t,0}\phi^x_{t/0,u_1}\phi^y_{t,u_2}} & z=0 \end{cases} \qquad (3.18)
\end{aligned}$$

We omit the superscript (a) in Eq. (3.18). Comparing Eq. (3.18) with Eq. (3.12) and (3.13), we find that the auxiliary distribution q we introduced above is actually equal to the posterior distribution over the topics on each link. Now we can write Q as:

$$\begin{aligned}
&Q(\theta,\rho,\phi|\theta^{(a)},\rho^{(a)},\phi^{(a)}) \\
=&\sum_{x,y,u_1,u_2}\sum_z p(z|x,y,u_1,u_2,\theta^{(a)},\rho^{(a)},\phi^{(a)},t)\log p(x,y|\theta)p(u_1|t/z,\phi)p(u_2|t/z,\phi) \\
=&\sum_{x,y,e^{x,y,t}_{i,j}\in\mathcal{E}^t_{x,y}} e^{x,y,t}_{i,j}\left(\sum_{z=1}^k q^{x,y,z(a)}_{i,j}\log\phi^x_{t/z,i}\phi^y_{t/z,j} + q^{x,y,0(a)}_{i,j}\log\phi^x_{t/0,i}\phi^y_{t,j}+\log\theta_{x,y}\right)
\end{aligned}$$

Maximization step (M-step). Find the parameters that maximize Q:

$$\theta^{(a+1)},\rho^{(a+1)},\phi^{(a+1)} = \arg\max_{\theta,\rho,\phi} Q(\theta,\rho,\phi|\theta^{(a)},\rho^{(a)},\phi^{(a)})$$

Using the Lagrange multiplier method, we can obtain the solution:

$$\theta_{x,y}^{(a+1)} = \frac{\sum_{e_{i,j}^{x,y,t}\in\mathcal{E}_{x,y}^t} e_{i,j}^{x,y,t}}{M^t} \tag{3.19}$$

$$\rho_{t,z}^{(a+1)} = \frac{\sum_{e_{i,j}^{x,y,t}\in\mathcal{E}_{x,y}^t} e_{i,j}^{x,y,t} q_{i,j}^{x,y,z(a)}}{M^t} \tag{3.20}$$

$$\phi_{t/z,i}^{x\;(a+1)} = \frac{\sum_{e_{i,j}^{x,y,t}\in\mathcal{E}_{x,y}^t} e_{i,j}^{x,y,t} q_{i,j}^{x,y,z(a)} + \sum_{e_{j,i}^{y,x,t}\in\mathcal{E}_{y,x}^t} e_{j,i}^{y,x,t} q_{j,i}^{y,x,z(a)}}{\sum_{e_{u,j}^{x,y,t}\in\mathcal{E}_{x,y}^t} e_{u,j}^{x,y,t} q_{u,j}^{x,y,z(a)} + \sum_{e_{j,u}^{y,x,t}\in\mathcal{E}_{y,x}^t} e_{j,u}^{y,x,t} q_{j,u}^{y,x,z(a)}} \tag{3.21}$$

$$\phi_{t/0,i}^{x\;(a+1)} = \frac{\sum_{e_{i,j}^{x,y,t}\in\mathcal{E}_{x,y}^t} e_{i,j}^{x,y,t} q_{i,j}^{x,y,0(a)}}{\sum_{e_{u,j}^{x,y,t}\in\mathcal{E}_{x,y}^t} e_{u,j}^{x,y,t} q_{u,j}^{x,y,0(a)}} \tag{3.22}$$

It is easy to verify the equivalence of Eq. (3.19)–(3.22) and Eq. (3.14)–(3.17). ■

This theorem shows that we can derive the same solution from both the unrolled generative model and the collapsed model. The unrolled model is natural and intuitive, but the collapsed model is easier for extension, as we will see in next subsection.

The theorem also incarnates q as a posterior distribution over the topics on each link. Based on it, we can calculate the expected number of topic-t/z links between every two nodes:

$$\hat{e}_{i,j}^{x,y,t/z} = e_{i,j}^{x,y,t} q_{i,j}^{x,y,z} \tag{3.23}$$

Then we have the update rules based on $\hat{e}_{i,j}^{x,y,z}$:

E-step:

$$\hat{e}_{i,j}^{x,y,t/z} = \frac{e_{i,j}^{x,y,t}\rho_{t,z}\phi_{t/z,i}^{x}\phi_{t/z,j}^{y}}{\sum_{c=1}^{k}\rho_{t,c}\phi_{t/c,i}^{x}\phi_{t/c,j}^{y} + \rho_{t,0}\phi_{t/0,i}^{x}\phi_{t,j}^{y}} \tag{3.24}$$

$$\hat{e}_{i,j}^{x,y,t/0} = \frac{e_{i,j}^{x,y,t}\rho_{t,0}\phi_{t/0,i}^{x}\phi_{t,j}^{y}}{\sum_{c=1}^{k}\rho_{t,c}\phi_{t/c,i}^{x}\phi_{t/c,j}^{y} + \rho_{t,0}\phi_{t/0,i}^{x}\phi_{t,j}^{y}} \tag{3.25}$$

M-step:

$$\theta_{x,y} = \frac{\sum_{v_i^x, v_j^y} e_{i,j}^{x,y,t}}{M^t} \tag{3.26}$$

$$\rho_{t,z} = \sum_{v_i^x, v_j^y} \frac{\hat{e}_{i,j}^{x,y,t/z}}{M^t} \tag{3.27}$$

$$\phi_{t/z,i}^x = \frac{\sum_{v_j^y} (\hat{e}_{i,j}^{x,y,t/z} + \hat{e}_{j,i}^{y,x,t/z})}{\sum_{v_u^x, v_j^y} (\hat{e}_{u,j}^{x,y,t/z} + \hat{e}_{j,u}^{y,x,t/z})} \tag{3.28}$$

$$\phi_{t/0,i}^x = \frac{\sum_{v_j^y} \hat{e}_{i,j}^{x,y,t/0}}{\sum_{v_u^x, v_j^y} \hat{e}_{u,j}^{x,y,t/0}} \tag{3.29}$$

These equations are intuitive. In E-step, the expected link weight of each subtopic $\hat{e}$ is calculated from the posterior distribution q given the current parameter estimates. This can be viewed as soft clustering of links. In M-step, the parameters are re-estimated based on the link clustering: the link type weight $\theta_{x,y}$ is calculated as dividing the total link weight of type (x, y) by the total link weight; the topic distribution ρ_t is estimated by the expected number of links in each subtopic; and the ranking distribution over nodes in each topic t/z is estimated by the total number of topic-t/z links associated these nodes.

We update $\hat{e}, \phi, \rho$ in each iteration because $\theta_{x,y}$ is a constant. The EM algorithm can be run multiple times with random initializations, and the solution with the best likelihood will be chosen.

The subnetwork for topic t/z is naturally extracted from the estimated $\hat{e}$ (expected link weight attributed to each topic). For efficiency purposes, we remove links whose weight is less than 1, and then filter out all resulting isolated nodes. We can then recursively apply the same generative model to the constructed subnetworks until the desired hierarchy is constructed.

3.2.2 Learning link type weights

The generative model described above does not differentiate between the importance of different link types. However, we may wish to discover topics that are biased towards certain types of links, and the bias may vary at different levels of the hierarchy. For example, in the computer science domain, the links between venues and other entities may be more important indicators than other link types in the top level of the hierarchy; however, these same links may be less useful for discovering subareas in the lower levels (*e.g.*, authors working in different subareas may publish in the same venue).

We therefore extend our model to capture the importance of different link types. We introduce a *link type weight* $\alpha_{x,y} > 0$ for each link type (x, y). We use these weights to scale a link's observed weight up or down, so that a unit-weight link of type (x, y) in the original network will have a *scaled* weight $\alpha_{x,y}$. Thus, a link of type (x, y) is valued more when $\alpha_{x,y} > 1$, less when $0 < \alpha_{x,y} < 1$, and becomes negligible as $\alpha_{x,y}$ approaches 0.

When the link type weights $\alpha_{x,y}$ are specified for our model, the EM inference algorithm is unchanged, with the exception that all the $e_{i,j}^{x,y,t}$ in the update equations should be replaced by $\alpha_{x,y}e_{i,j}^{x,y,t}$. When all $\alpha_{x,y}$'s are equal, the weight-learning model reduces to the basic model. Most of the time, the weights of the link types will not be specified explicitly by users, and must therefore be learned from the data.

We first note an important property of our model, justifying our previous claim that link weights need not be integers.

Lemma 3.1 (Scale-invariant) *The EM solution is invariant to a constant scaleup of all the link weights. That is, if we replace all the* $e_{i,j}^{x,y,t}$ *with* $ce_{i,j}^{x,y,t}$*, the resulting* $q_{i,j}^{x,y,z}, \rho_{t,z}, \theta_{x,y}$ *and* $\phi_{t/z,i}^{x}$ *all remain unchanged for topic* t *and all descendant topics of* t.

The proof is straightforward by induction.

With the scale-invariant property on the link weights, we can prove the following theorem.

Theorem 3.2 *For a set of* l *positive numbers* $\alpha_{x,y} > 0$*, there exist another set of* l *positive numbers* $\beta_{x,y} > 0$*, such that the EM solution based on link weights* $\alpha_{x,y}$ *and* $\beta_{x,y}$ *are identical, and* $\prod_{e_{i,j}^{x,y,t}>0} e_{i,j}^{x,y,t} = \prod_{e_{i,j}^{x,y,t}>0}(\beta_{x,y}e_{i,j}^{x,y,t})$.

Proof: Let $\chi = \dfrac{\prod_{e_{i,j}^{x,y,t}>0} e_{i,j}^{x,y,t}}{\prod_{e_{i,j}^{x,y,t}>0}(\alpha_{x,y}e_{i,j}^{x,y,t})}$, $N = \sum_{x,y} n_{x,y}$. We define:

$$\beta_{x,y} \equiv \chi^{\frac{1}{N}}\alpha_{x,y} \tag{3.30}$$

The scale-invariant property implies that the EM solution based on link weights $\alpha_{x,y}$ and $\beta_{x,y}$

are identical. So we have:

$$\begin{aligned}
&\prod_{e_{i,j}^{x,y,t}>0} (\beta_{x,y} e_{i,j}^{x,y,t}) = \prod_{e_{i,j}^{x,y,t}>0} (\chi^{\frac{1}{N}} \alpha_{x,y} e_{i,j}^{x,y,t}) \\
=&\chi \prod_{e_{i,j}^{x,y,t}>0} (\alpha_{x,y} e_{i,j}^{x,y,t}) = \prod_{e_{i,j}^{x,y,t}>0} e_{i,j}^{x,y,t} \qquad (3.31)
\end{aligned}$$

■

With this theorem, we can assume that *w.l.o.g.*, the product of all the non-zero link weights remains invariant before and after scaling:

$$\prod_{e_{i,j}^{x,y,t}>0} e_{i,j}^{x,y,t} = \prod_{e_{i,j}^{x,y,t}>0} (\alpha_{x,y} e_{i,j}^{x,y,t}) \qquad (3.32)$$

which reduces to $\prod_{x,y} \alpha_{x,y}^{n_{x,y}} = 1$, where $n_{x,y} = |\mathcal{E}_{x,y}^t|$ is the number of non-zero links with type (x, y). With this contraint, we maximize the likelihood $p(\{e_{i,j}^{x,y,t}\}|\theta, \rho, \phi, \alpha)$:

$$\max \prod_{v_i^x, v_j^y} \frac{(\alpha_{x,y} M_{x,y}^t s_{i,j}^{x,y,t})^{\alpha_{x,y} e_{i,j}^{x,y,t}} \exp(-\alpha_{x,y} M_{x,y}^t s_{i,j}^{x,y,t})}{(\alpha_{x,y} e_{i,j}^{x,y,t})!} \qquad (3.33)$$

$$s.t. \prod_{x,y} \alpha_{x,y}^{n_{x,y}} = 1, \alpha_{x,y} > 0 \qquad (3.34)$$

where $M_{x,y}^t = \sum_{v_i^x, v_j^y} e_{i,j}^{x,y,t}$ is the total weight for type (x, y) links. With Stirling's approximation $n! \sim (\frac{n}{e})^n \sqrt{2\pi n}$, we rewrite the log likelihood:

$$\begin{aligned}
\max \sum_{v_i^x, v_j^y} \Big(&\alpha_{x,y} e_{i,j}^{x,y,t} \log(\alpha_{x,y} M_{x,y}^t s_{i,j}^{x,y,t}) - \alpha_{x,y} M_{x,y}^t s_{i,j}^{x,y,t} \qquad (3.35) \\
&-\alpha_{x,y} e_{i,j}^{x,y,t} [\log(\alpha_{x,y} e_{i,j}^{x,y,t}) - 1] - \frac{1}{2} \log(\alpha_{x,y} e_{i,j}^{x,y,t}) \Big)
\end{aligned}$$

$$s.t. \sum_{x,y} n_{x,y} \log \alpha_{x,y} = 0 \qquad (3.36)$$

Using the Lagrange multiplier method, we can find the optimal value for α when the other parameters are fixed:

$$\alpha_{x,y} = \frac{\left[\prod_{x,y}\left(\frac{1}{n_{x,y}}\sum_{i,j} e_{i,j}^{x,y,t}\log\frac{e_{i,j}^{x,y,t}}{M_{x,y}^{t}s_{i,j}^{x,y,t}}\right)^{n_{x,y}}\right]^{\frac{1}{\sum_{x,y} n_{x,y}}}}{\frac{1}{n_{x,y}}\sum_{i,j} e_{i,j}^{x,y,t}\log\frac{e_{i,j}^{x,y,t}}{M_{x,y}^{t}s_{i,j}^{x,y,t}}} \tag{3.37}$$

With some transformation of the denominator:

$$\begin{aligned}\sigma_{x,y} &= \frac{1}{n_{x,y}}\sum_{i,j} e_{i,j}^{x,y,t}\log\frac{e_{i,j}^{x,y,t}}{M_{x,y}^{t}s_{i,j}^{x,y,t}} \\ &= \frac{M_{x,y}^{t}}{n_{x,y}}\sum_{v_i^x,v_j^y}\frac{e_{i,j}^{x,y,t}}{M_{x,y}^{t}}\log\frac{e_{i,j}^{x,y,t}/M_{x,y}^{t}}{s_{i,j}^{x,y,t}}\end{aligned} \tag{3.38}$$

we can see more clearly that the link type weight is negatively correlated with two factors: the average link weight $\frac{M_{x,y}^{t}}{n_{x,y}}$ and the KL-divergence of the expected link weight distribution to the observed link weight distribution $\sum_{v_i^x,v_j^y}\frac{e_{i,j}^{x,y,t}}{M_{x,y}^{t}}\log\frac{e_{i,j}^{x,y,t}/M_{x,y}^{t}}{s_{i,j}^{x,y,t}}$. The first factor is used to balance the scale of link weights of different types (*e.g.*, a type-1 link always has X times greater weight than a type-2 link). The second factor measures the importance of a link type in the model. The more the prediction diverges from the observation, the worse the quality of a link type.

So we have the following iterative algorithm for optimizing the joint likelihood:

1. Initialize all the parameters.

2. Fixing α, update ρ, ϕ using EM equations:

$$\hat{e}_{i,j}^{x,y,z} = \frac{\alpha_{x,y} e_{i,j}^{x,y,t} \rho_{t,z} \phi_{t/z,i}^{x} \phi_{t/z,j}^{y}}{\sum_{c=1}^{k} \rho_{t,c} \phi_{t/c,i}^{x} \phi_{t/c,j}^{y} + \rho_{t,0} \phi_{t/0,i}^{x} \phi_{t,j}^{y}} \tag{3.39}$$

$$\hat{e}_{i,j}^{x,y,t/0} = \frac{\alpha_{x,y} e_{i,j}^{x,y,t} \rho_{t,0} \phi_{t/0,i}^{x} \phi_{t,j}^{y}}{\sum_{c=1}^{k} \rho_{t,c} \phi_{t/c,i}^{x} \phi_{t/c,j}^{y} + \rho_{t,0} \phi_{t/0,i}^{x} \phi_{t,j}^{y}} \tag{3.40}$$

$$\rho_{t,z} = \frac{\sum_{v_i^x, v_j^y} \hat{e}_{i,j}^{x,y,t/z}}{\sum_{x,y} \alpha_{x,y} M_{x,y}^{t}} \tag{3.41}$$

$$\phi_{t/z,i}^{x} = \frac{\sum_{v_j^y} (\hat{e}_{i,j}^{x,y,t/z} + \hat{e}_{j,i}^{y,x,t/z})}{\sum_{v_u^x, v_j^y} (\hat{e}_{u,j}^{x,y,t/z} + \hat{e}_{j,u}^{y,x,t/z})} \tag{3.42}$$

$$\phi_{t/0,i}^{x} = \frac{\sum_{v_j^y} \hat{e}_{i,j}^{x,y,t/0}}{\sum_{v_u^x, v_j^y} \hat{e}_{u,j}^{x,y,t/0}} \tag{3.43}$$

3. Fixing ρ, ϕ, update α using Eq. (3.37).

4. Repeat steps 2 and 3 until the likelihood converges.

In each iteration, the time complexity is $\mathcal{O}(\sum_{x,y} n_{x,y})$, *i.e.*, linear to the total number of non-zero links. The likelihood is guaranteed to converge to a local optimum. Once again, a random initialization strategy can be employed to choose a solution with the best local optimum.

It is important to note that the learned link weights indicate the overall importance of a type in the following sense: how much the original link weight of each type should be rescaled. A larger link weight corresponds to a link type that should be counted more when we fit the the likelihood of the observation. If we want to see the subtle difference of the importance for each individual link, a possible modeling strategy is to parameterize the link weights $\alpha_{x,y}$, *e.g.*, according to the attributes or topological features of nodes such as their degree or weighted degree. It can be considered for future work.

3.2.3 Shape of hierarchy

In this section, we discuss the following issues that affect the shape of the constructed hierarchy.

Number of children for each topic. Since our framework is recursive, the shape of the tree is essentially determined by how many children each node has, *i.e.*, how many subtopics each topic has. For every topic, our model can work with an arbitrary number of subtopics that is larger than 1. However, it may be more reasonable to have a certain number of subtopics than others. In general, we prefer each topic to have a small number of subtopics, *e.g.*, between 2 and 10, in order to make it easy for browsing. For example, if the root has 5 subtopics and each of them has 4 subtopics, the three-level hierarchy is in general easier to browse than directly showing all 20 topics under the root.

Given a range of subtopic number, such as $[2, 10]$, we would like to choose a reasonable number of children k for each topic. It is a model selection problem. Among various model selection strategies in the literature, we select two of them and introduce how they can be adapted for our model.

The first strategy was proposed by Smyth [75] to adopt cross-validation to choose the parameter K. In our setting, we can first fit the generative model to a sampled subnetwork H^t of the given network G^t. Then we evaluate the likelihood of the model on the rest part of the network $G^t - H^t$, which is called the held-out network. By checking the averaged held-out likelihood with varying number of sub-clusters, the parameter with the maximum value will be chosen as the best candidate.

The second strategy is based on the Bayesian information criterion (BIC). A similar criterion is Akaike information criterion (AIC). Both BIC and AIC resolve the overfitting problem. When we increase the number of topics k, it is possible to increase the likelihood, but may result in overfitting because the model will have a larger number of parameters. BIC and AIC introduce a penalty term for the number of parameters in the model, and the penalty term is larger in BIC than in AIC. Using BIC, the measure for our model is defined as:

$$BIC = -2 \log p \left(\{e_{i,j}^{x,y,t}\} | \theta, \rho, \phi \right) + |\theta, \rho, \phi| \cdot \log |\mathcal{E}^t|$$

where $|\theta, \rho, \phi|$ is the number of free parameters in the model and $|\mathcal{E}^t|$ refers to the size of observed links. As we only care about the topic number k, $|\theta, \rho, \phi|$ can be reduced to $|\mathcal{V}^t|k$ plus a constant independent of k, where $|\mathcal{V}^t|$ is the number of nodes. We can then select k with the largest BIC score.

BIC is derived under the assumptions that the data distribution is in the exponential family. Cross validation only assumes that the sampled network and the held-out network are generated from the same model. Comparing these two criteria, we generally recommend cross validation over BIC when there are sufficient data. However, when the network is small, cross validation is prone to high variation and BIC can be used as an alternative.

Depth of the hierarchy. A simple and intuitive strategy to decide the depth of the hierarchy is to rely on the selected number of children mentioned in above. For example, if the best number of topics is $k = 1$, it implies we should stop expanding the current topic node. In practice, we can set a threshold on the largest depth of the tree, as well as the size of the network. Once the tree has reached the maximal depth, or the size of the network in current topic is too small, we can cease the recursion. A general implication is that the more children each node has, the less deep the final hierarchy will be.

Balance of subtree size. The distribution of ρ_t's determines the size of subtrees. The more evenly distributed are ρ_t's, the more balanced are the subtrees. Generally we would like to generate a balanced tree because it is efficient for browsing. If this is the case, we should randomly initialize the topic of each link from a uniform distribution. In case one would like to generate a skewed hierarchy, the random initialization of each link's topic distribution should follow a non-uniform multinomial, which can be generated from a Dirichlet prior. Our model can be extended into a Bayesian framework, which can incorporate conjugate prior for all the parameters. The shape of the hierarchy can then be controled by the hyperparameters of prior. It is left as future work.

3.3 Experiments

Lack of gold standard is a known issue for unsupervised topic modeling methods. As such, people have proposed evaluation metrics without relying on labels. Our task of constructing multi-typed topic hierarchy is new and there is neither gold standard for it. We leverage the existing evaluation metrics, pointwise mutual information [66] and intrusion detection [17] that are proved to be effective in text-based topic modeling, and modify them to evaluate our multi-typed topic hierarchy. The metrics can be used to compare different methods in arbitrary datasets.

We evaluate the performance of our proposed method on two datasets (see Table 3.4 for summary statistics of the constructed networks):

• **DBLP**. We collected 33,313 recently published computer science papers from DBLP[2]. We constructed a heterogeneous network with three node types: term (from paper title), author and venue, and 5 link types: term-term, term-author, term-venue, author-author and author-venue.[3]

• **NEWS**. We crawled 43,168 news articles on 16 top stories from Google News,[4] extracted text content from html pages by heuristic rules, and ran an information extraction algorithm [52] to extract entities. We constructed a heterogeneous network with three node types: term (from article title), person and location, and 6 link types: term-term, term-person, term-location, person-person, person-location and location-location.

The evaluation is twofold: i) we evaluate the efficacy of subtopic discovery given a topic and its associated heterogeneous network; and ii) we perform several 'intruder detection' tasks to evaluate the quality of the constructed hierarchy based on human judgment.

3.3.1 Efficacy of subtopic discovery

We first present a set of experiments designed to evaluate just the subtopic discovery step (Step 2 in Section 3.2).

Evaluation measure. We extend the pointwise mutual information (PMI) metric in order to measure the quality of the multi-typed topics. The metric of pointwise mutual information PMI has been proposed by [66] as a way of measuring the semantic coherence of topics. It is generally preferred over other quantitative metrics such as perplexity or the likelihood of held-out data [79]. In order to measure the quality of our multi-typed topics, we extend the definition of PMI as follows:

For each topic, PMI calculates the average relatedness of each pair of the words ranked at top-K:

$$PMI(\mathbf{v}, \mathbf{v}) = \frac{2}{K(K-1)} \sum_{1 \leq i < j \leq K} \log \frac{p(v_i, v_j)}{p(v_i)p(v_j)} \tag{3.44}$$

where $PMI \in [-\infty, \infty]$, and $\mathbf{v}$ are the top K most probable words of the topic. $PMI = 0$ implies that these words are independent; $PMI > 0$ (< 0) implies they are overall positively (negatively)

[2]We chose papers published in 20 conferences related to the areas of Artificial Intelligence, Databases, Data Mining, Information Retrieval, Machine Learning, and Natural Language Processing from `http://www.dblp.org/`

[3]As a paper is always published in exactly one venue, there can naturally be no venue-venue links.

[4]The 16 topics chosen were: Bill Clinton, Boston Marathon, Earthquake, Egypt, Gaza, Iran, Israel, Joe Biden, Microsoft, Mitt Romney, Nuclear power, Steve Jobs, Sudan, Syria, Unemployment, US Crime.

correlated.

However, our multi-typed topic contains not only words, but also other types of entities. So we define *heterogeneous* pointwise mutual information as:

$$HPMI(\mathbf{v^x}, \mathbf{v^y}) = \begin{cases} \frac{2}{K(K-1)} \sum_{1 \le i < j \le K} \log \frac{p(v_i^x, v_j^y)}{p(v_i^x)p(v_j^y)} & x = y \\ \frac{1}{K^2} \sum_{1 \le i,j \le K} \log \frac{p(v_i^x, v_j^y)}{p(v_i^x)p(v_j^y)} & x \neq y \end{cases} \tag{3.45}$$

where $\mathbf{v^x}$ are the top K most probable type-x nodes in the given topic. When $x = y$, HPMI reduces to PMI. The HPMI-score for every link type (x, y) is calculated and averaged to obtain an overall score. We set $K = 20$ for all node types.[5]

Methods for comparison:

• **CATHYHIN (equal weight)** – The weight for every link type is set to be 1.

• **CATHYHIN (learn weight)** – The weight of each link type is learned, as described in Section 3.2.2. No parameters need hand tuning.

• **CATHYHIN (norm weight)** – The weight of each link type is explicitly set as: $\alpha_{x,y} = \frac{1}{\sum_{i,j} e_{i,j}^{x,y}}$. This is a heuristic normalization which forces the total weight of the links for each link type to be equal.

• **NetClus** – The current state-of-the-art clustering and ranking method for heterogeneous networks. We use the implementation by [25]. The smoothing parameter λ_S is tuned by a grid search in $[0, 1]$. The optimal value for these two domains is 0.3 and 0.5 respectively. Note that the link type weight learning method for CATHYHIN does not apply to NetClus because NetClus does not have a single objective function to optimize.

• **TopK** – Select the top K nodes from each type according to their frequency to form a pseudo topic. This method serves as a baseline value for the proposed HPMI metric.

Experiment setup. We discover the subtopics of four datasets:

• DBLP (20 conferences) – Aforementioned DBLP dataset. This dataset is used for evaluating the performance when constructing the first level of the hierarchy.

• DBLP (database area) – A subset of the DBLP dataset consisting only of papers published in 5 Database conferences. By using this dataset, which roughly repesents a subtopic of the full DBLP

[5]The one exception is venues, as there are only 20 venues in the DBLP dataset, so we set $K = 3$ in this case.

dataset, we analyze the quality of discovered subtopics in a lower level of the hierarchy.

• NEWS (16 topics) – Aforementioned NEWS dataset.

• NEWS (4 topic subset) – A subset of the NEWS dataset limited to 4 topics, which center around different types of entities: Bill Clinton, Boston Marathon, Earthquake, Egypt.

We use the BIC model selection criterion described in Section 3.2.3 to select k. It aligns with our prior knowledge. For example, on DBLP (20 conferences), $k = 6$ and there are 6 actual areas in the data.

Experiment results. All the methods finish in 1.5 hours for these datasets, on a Windows server running MATLAB R2011a with Intel Xeon X5650 2.67GHz and 48GB RAM.

We show the heterogeneous pointwise mutual information averaged over the learned topics in Tables 3.2 and 3.3, our generative model consistently posts a higher HPMI score than NetClus (and TopK) across all links types in every dataset. Although NetClus HPMI values are better than the TopK baseline, the improvement of our best performing method - CATHYHIN (learn weight) - over the TopK baseline are better than the improvement posted by NetClus by factors ranging from 2 to 5.8. Even the improvement over the TopK baseline of CATHYHIN (equal weight), which considers uniform link type weights, is better than the improvement posted by NetClus by factors ranging from 1.6 to 4.6.

CATHYHIN with learned link type weights consistently yields the highest overall HPMI scores, although CATHYHIN with normalized link type weights sometimes shows a slightly higher score for particular link types (*e.g.*, Author-Author for both DBLP datasets, and Person-Person for both NEWS datasets). CATHYHIN (norm weight) assigns a high weight to a link type whose total link weights were low in the originally constructed network, pushing the discovered subtopics to be more dependent on that link type. Normalizing the link type weights does improve CATHYHIN performance in many cases, as compared to using uniform link type weights. However, this heuristic determines the link type weight based solely on their link density. It can severely deteriorate the coherence of desne but valuable link types, such as Term-Term in both DBLP datasets, and rely too heavily on sparse but uninformative entities, such as Venues in the Database subtopic of the DBLP dataset.

Figure 3.8 demonstrates the learned link weights by CATHYHIN (learn weight) on DBLP datasets. At the first level, the term-venue and author-venue link types are assigned high weight, because the venue is a most important discriminator for general areas (Artificial Intelligence, Databases, Data Mining, Information Retrieval, Machine Learning, and Natural Language Processing). At the second level, the venue links are much less useful for discovering subtopics in each area.

We may conclude from these experiments that CATHYHIN's unified generative model consistently outperforms the state-of-the-art heterogeneous network analysis technique NetClus. In order to generate coherent, multi-typed topics at each level of a topical hierarchy, it is important to learn the optimal weights of different entity types, which depends on the link type density, the granularity of the topic to be partitioned, and the specific domain.

3.3.2 Topical hierarchy quality

The second set of evaluations assesses the ability of our method to construct a hierarchy of multi-typed topics that human judgement deems to be high quality. We generate and analyze multi-typed topical hierarchies using the DBLP dataset (20 conferences) and the NEWS dataset (16 topics collection).

Experiment setup. We design three tasks inspired by [17], who were the first to explore human evaluation of topic models. Each task involves a set of questions asking humans to discover the 'intruder' object from several options. Three annotators manually completed each task, and their evaluation scores were pooled.

The first task is Phrase Intrusion, which evaluates how well the hierarchies are able to separate phrases in different topics. Each question consists of X ($X = 5$ in our experiments) phrases; $X - 1$ of them are randomly chosen from the top phrases of the same topic and the remaining phrase is randomly chosen from a sibling topic. The second task is Entity Intrusion, a variation that evaluates how well the hierarchies are able to separate entities present in the dataset in different topics. For each entity type, each question consists of X entity patterns; $X - 1$ of them are randomly chosen from the top patterns of the same topic and the remaining entity pattern is randomly chosen from a sibling topic. This task is constructed for each entity type in each dataset (Author and Venue in DBLP; Person and Location in NEWS). The third task is Topic Intrusion, which tests the quality of the parent-child relationships in the generated hierarchies. Each question consists of a parent topic t and X candidate child topics. $X - 1$ of the child topics are actual children of t in the generated hierarchy, and the remaining child topic is not. Each topic is represented by its top 5 ranked patterns of each type - *e.g.*, for the NEWS dataset, the top 5 phrases, people, and locations are shown for each topic.

For DBLP, we generate 210 phrase intrusion questions, 210 entity instrusion questions for both author and venue type, and 60 topic intrusion questions. For NEWS, we generate 280 phrase intrusion questions, 280 entity instruction questions for both person and location type, and 100 topic intrusion questions. Figure 3.9 shows examples of generated questions in DBLP. For each question, 3 human annotators with background knowledge of computer science and news select the intruder phrase, entity, or subtopic. If they are unable to make a choice, or choose incorrectly or inconsistently, the question is marked as a failure.

Methods for comparison:

• **CATHYHIN** – As defined in Section 3.2.

• **CATHYHIN$_\mathbf{1}$** – The pattern length of text and every entity type is restricted to 1.

• **CATHY** – As defined in Section 3.1, the hierarchy is constructed only from textual information.

• **CATHY$_\mathbf{1}$** – The phrase length is restricted to 1.

• **CATHY$_\mathbf{heuristic\_HIN}$** – Since neither CATHY nor CATHY$_1$ provides topical ranks for entities, we construct this method to have a comparison for the Entity Intrusion task. We use a heuristic entity ranking method based on the textual hierarchy generated by CATHY, and the original links in the network (see Chapter 5)..

• **NetClus$_\mathbf{phrase}$** – NetClus is used for subtopic discovery, followed by the topical mining and ranking method of CATHYHIN, as described in Chapter 4.

• **NetClus$_\mathbf{phrase\_1}$** – Equivalent to NetClus$_{\text{phrase}}$ with the phrase length restricted to 1.

• **NetClus** – As defined in [78].

The optimal smoothing parameter for NetClus is $\lambda_S = 0.3$ and 0.7 in DBLP and NEWS respectively.

Table 3.5 displays the results of the intruder detection tasks. For the Entity Intrusion task on the DBLP dataset, we restricted the entity pattern length to 1 in order to generate meaningful questions. This renders the methods CATHYHIN$_1$ and NetClus$_{\text{phrase\_1}}$ equivalent to CATHYHIN and NetClus$_{\text{phrase}}$ respectively, so we omit the former methods from reporting.

Experiment results. The Phrase Intrusion task performs much better when phrases are used rather than unigrams, for both CATHYHIN and CATHY, on both datasets. The NEWS dataset exhibits a stronger preference for phrases, as opposed to the DBLP dataset, which may be due to the fact that the terms in the NEWS dataset are more likely to be noisy and uninformative outside of their context, whereas the DBLP terms are more technical and therefore easier to interpret. This characteristic may also help explain why the performance of every method on DBLP data is consistently higher than on NEWS data. However, neither phrase mining and ranking nor unigram ranking can make up for poor performance during the topic discovery step, as seen in the three NetClus variations. Therefore, both phrase representation and high quality topics are necessary for good topic interpretability.

For the Entity Intrusion task, all of the relevant methods show comparable performance in identifying Author and Venue intruders in the DBLP dataset (though CATHYHIN is still consistently the highest). Since the DBLP dataset is well structured, and the entity links are highly trustworthy, identifying entities by topic is likely easier. However, the entities in the NEWS dataset were automatically discovered from the data, and the link data is therefore noisy and imperfect. CATHYHIN is the most effective in identifying both Location and Person intruders. Once again, both better topic discovery and improved pattern representations are responsible for CATHYHIN's good results, and simply enhancing the pattern representations, whether for CATHY or NetClus, cannot achieve competitive performance.

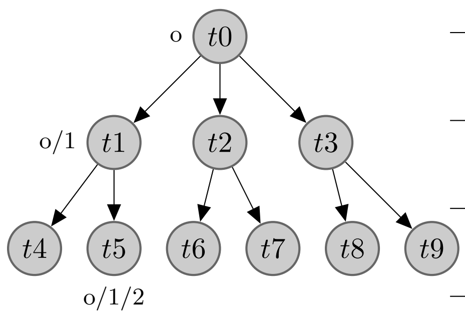
CATHYHIN performs very well in the Topic Intrusion task on both datasets. Similar to the Phrase Intrusion task, both CATHYHIN and CATHY yield equally good or better result when phrases and entity patterns are mined, rather than just terms and single entities. The fact that CATHYHIN always outperforms CATHY demonstrates that utilizing entity link information is indeed helpful for improving topical hierarchy quality. In all three intruder detection tasks on both datasets, CATHYHIN consistently outperforms all other methods, showing that an integrated heterogeneous model consistently produces a more robust hierarchy which is more easily interpreted by human judgment.

3.3.3 Case study

In Section 3.3.2, we analyzed the two main reasons for the performance gain of CATHYHIN: improved pattern representations and better topic discovery. In Figure 3.4 we have shown real examples of the constructed hierarchy in the DBLP data. It is clear that the multi-typed entities enrich the context of each topic and improve the representations of text-only topics. Here we use one simple example to illustrate the topic discovery performance, using the same topic representations.

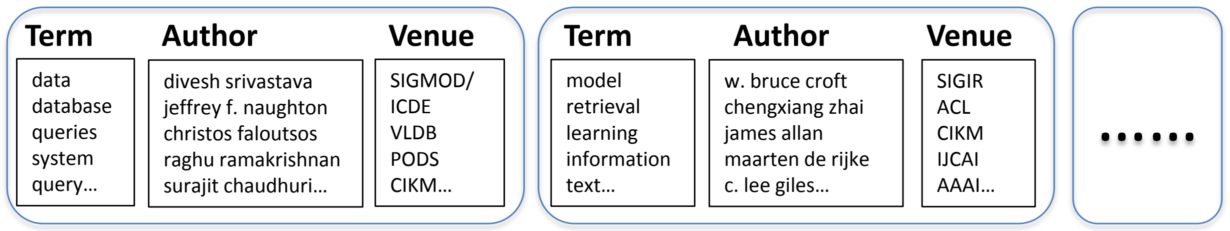
Table 3.6 illustrates three representations of the topic 'information retrieval' (one of the 6 areas in DBLP dataset). Overall, CATHYHIN finds more 'pure' information retrieval entities. $\text{CATHY}_{\text{heuristic\_HIN}}$ generates very similar top-ranked phrases and venues, but different authors. While the top authors found by $\text{CATHY}_{\text{heuristic\_HIN}}$ indeed work on information retrieval, they spread their interest in other fields too, such as data mining and human computer interaction. This is because $\text{CATHY}_{\text{heuristic\_HIN}}$ only uses the links between text and entities to rank entities posterior to the text-based topic discovery, while CATHYHIN can further use the author-author and author-venue links to refine the topics and find the more accurate position for each entity. $\text{NetClus}_{\text{pattern}}$ also utilizes the multiple types of links, but it mixes different topics, such as information retrieval and natural language processing, perhaps due to the hard partitioning of papers and heuristic combination of ranking and clustering.

As a worst-case study, Table 3.7 illustrates three representations of the topic 'Egypt' (one of the 16 top stories in NEWS dataset), each with its least comprehensible subtopic. The locations found within the CATHYHIN subtopic are sensible. However, $\text{CATHY}_{\text{heuristic\_HIN}}$ first constructs phrase-represented topics from text, and then uses entity link information to rank entities in each topic. Thus the entities are not assured to fit well into the constructed topic, and indeed, the $\text{CATHY}_{\text{heuristic\_HIN}}$ subtopic's locations are not reasonable given the parent topic. For example, CATHY discovers *'supreme leader/army general sex/court/supreme court/egypts prosecutor general'* to be a subtopic of *'egypt/egypts morsi/egypt imf loan/egypts president/muslim brotherhood,'*. Resorting to the original network links to discover each topic's entity rankings results in the claim that the locations *'US/Sudan/Iran/Washington'* represent a subtopic of locations *'Egypt/Cairo/Tahrir Square/Port Said,'* which is not easily interpretable. Finally, $\text{NetClus}_{\text{pattern}}$ conflates 'Egypt' with several other topics, and the pattern representations can do little to improve



Topic	Word distribution	Representative phrase
t0(o)	data:0.01, learning:0.01 . . .	database system, machine learning . . .
t1(o/1)	database:0.05, system:0.01 . . .	database system, management . . .
t2(o/2)	information:0.1, retrieval:0.05 . . .	information retrieval, web search . . .
t3(o/3)	learning:0.11, classification:0.01 . . .	learning, classification, feature selection . . .
t4(o/1/1)	query:0.12, processing:0.07 . . .	query processing, query optimization . . .
t5(o/1/2)	system:0.08, distributed:0.03 . . .	distributed database, concurrency control . . .

Figure 3.1: An example of the topical hierarchy. Each topic can be denoted by the path from root topic to it



Term	Author	Venue
data	divesh srivastava	SIGMOD/ICDE
database	jeffrey f. naughton	VLDB
queries	christos faloutsos	PODS
system	raghu ramakrishnan	CIKM...
query...	surajit chaudhuri...	

Term	Author	Venue
model	w. bruce croft	SIGIR
retrieval	chengxiang zhai	ACL
learning	james allan	CIKM
information	maarten de rijke	IJCAI
text...	c. lee giles...	AAAI...

Figure 3.2: Sample output from NetClus [78] – clusters of multi-typed entities. Each rounded rectangle represents one cluster, containing a ranked list of unigrams and two ranked lists of entities

the topic interpretability.

Figures and tables

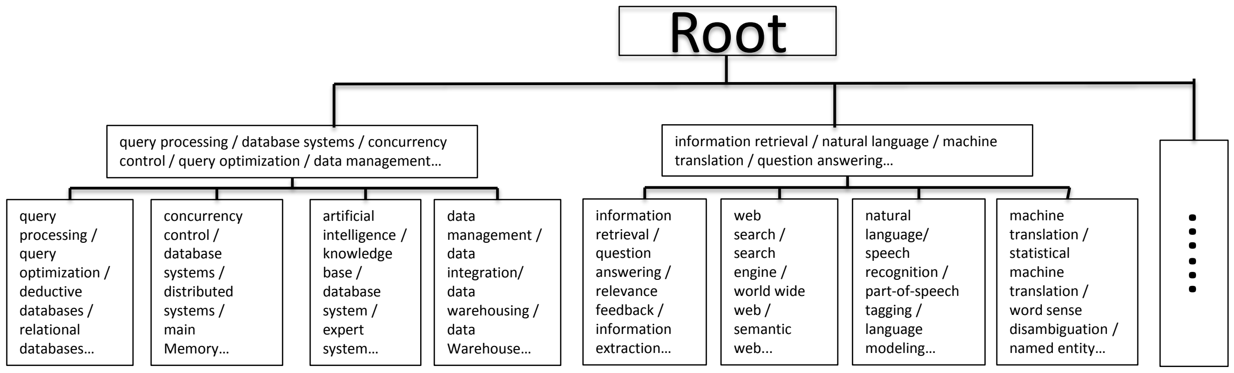

Root
query processing / database systems / concurrency control / query optimization / data management…
information retrieval / natural language / machine translation / question answering…
query processing / query optimization / deductive databases / relational databases…
concurrency control / database systems / distributed systems / main Memory…
artificial intelligence / knowledge base / database system / expert system…
data management / data integration/ data warehousing / data Warehouse…
information retrieval / question answering / relevance feedback / information extraction…
web search / search engine / world wide web / semantic web…
natural language/ speech recognition / part-of-speech tagging / language modeling…
machine translation / statistical machine translation / word sense disambiguation / named entity…

Figure 3.3: Sample output from CATHY [88] – topical hierarchy of text only. Each node in the hierarchy contains a ranked list of phrases

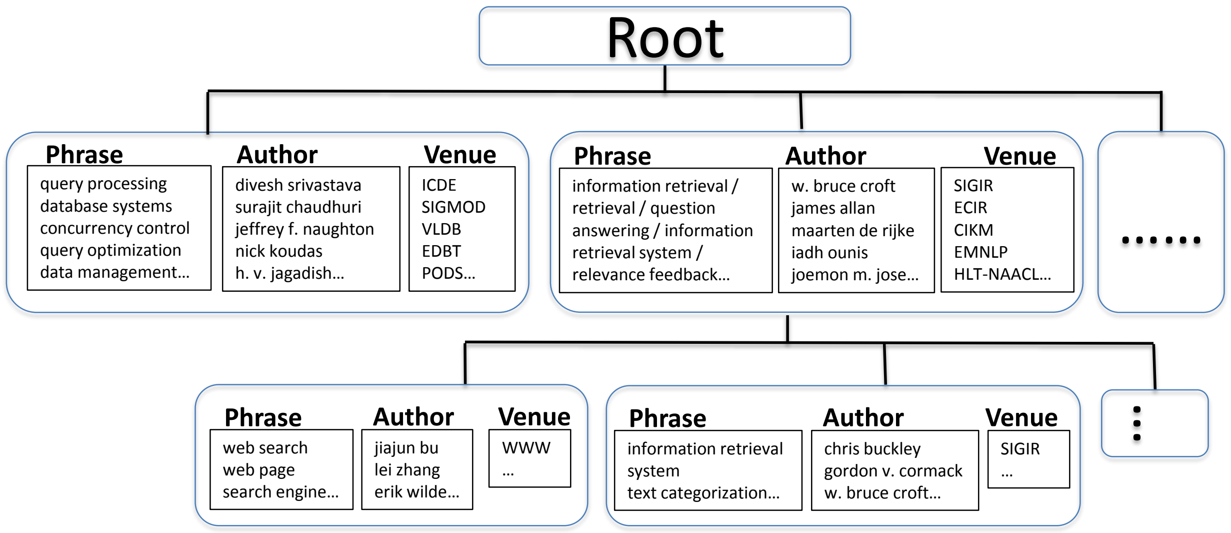


Figure 3.4: Sample output from CATHYHIN – topical hierarchy of multi-typed entities. Each node has a ranked list of phrases and two ranked entity lists

Table 3.1: Notations used in our model

Symbol	Description
G^t	the edge-weighted network associated with topic t
$\mathcal{V}_x^t$	the set of nodes of type x in topic t
$\mathcal{E}_{x,y}^t$	the set of non-zero link weights of type (x, y) in topic t
π_t	the parent topic of topic t
C_t	the number of child topics of topic t
o	the root topic
m	the number of node types
l	the number of link types
$n_{x,y}$	the total number of type-x and type-y node pairs that have links
v_i^x	the i-th node of type x
$e_{i,j}^{x,y,t}$	the link weight between v_i^x and v_j^y in topic t
M_t	the sum of link weight in topic t: $\sum_{i,j,x,y} e_{i,j}^{x,y,t}$
$M_t^{x,y}$	the sum of type-(x, y) link weight in topic t: $\sum_{i,j} e_{i,j}^{x,y,t}$
ϕ_t^x	the distribution over type-x nodes in topic t
$\phi_{t/0}^x$	the background distribution over type-x nodes in topic t
ρ	the distribution over subtopics
θ	the distribution over link types
$\alpha_{x,y}$	the importance of link type (x, y)

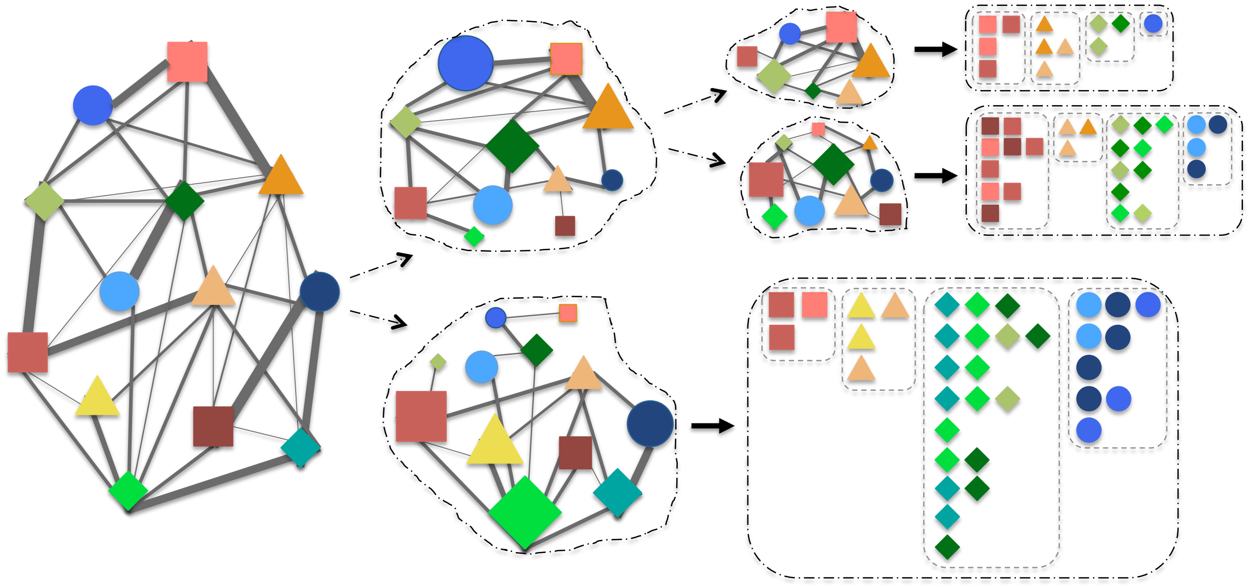


Figure 3.5: An illustration of the CATHYHIN framework. **(L)** Step 1: CATHYHIN analyses a node-typed and edge-weighted network, with no central star objects. **(M)** Step 2: A unified generative model is used to partition the edge weights into clusters and rank single nodes in each cluster (here, node rank within each node type is represented by variations in node size). **(R bottom)** Step 3: Patterns of nodes are ranked within each cluster, grouped by type. This step is discussed in Chapter 4. **(R top)** Step 4: Each cluster is also an edge-weighted network, and is therefore recursively analyzed. The final output is a hierarchy, where the patterns of nodes of each cluster have a ranking within that cluster, grouped by type

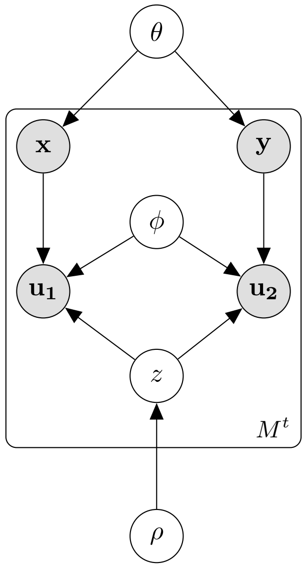


Figure 3.6: The generative process of the 'unit-weight' links

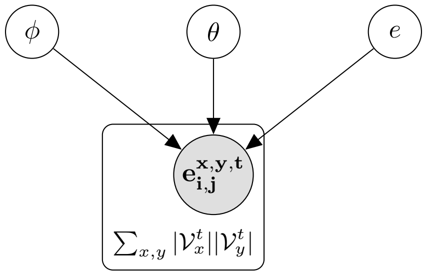

ϕ
θ
e
$\mathbf{e_{i,j}^{x,y,t}}$
$\sum_{x,y} |\mathcal{V}_x^t||\mathcal{V}_y^t|$

Figure 3.7: The 'collapsed' generative process of the link weights

Table 3.2: Heterogeneous pointwise mutual information in DBLP (20 Conferences and Database area)

DBLP (Database Area)	Term-Term	Term-Author	Author-Author	Term-Venue	Author-Venue	Overall
TopK	-0.5228	-0.1069	0.4545	0.0348	-0.3650	-0.0761
NetClus	-0.3962	0.0479	0.4337	0.0368	-0.2857	0.0260
CATHYHIN (equal weight)	0.0561	0.4799	0.6496	0.0722	-0.0033	0.3994
CATHYHIN (norm weight)	-0.1514	0.3816	**0.6971**	0.0408	**0.2464**	0.3196
CATHYHIN (learn weight)	**0.3027**	**0.6435**	0.5574	**0.1165**	0.1805	**0.5205**
DBLP (20 Conferences)	Term-Term	Term-Author	Author-Author	Term-Venue	Author-Venue	Overall
TopK	-0.4825	-0.0204	0.5466	-1.0051	-0.4208	-0.0903
NetClus	-0.1995	0.5186	0.5404	0.2851	1.2659	0.4045
CATHYHIN (equal weight)	0.2936	0.8812	0.6595	0.5191	1.0466	0.6949
CATHYHIN (norm weight)	0.1825	0.8674	**0.9476**	0.7472	1.3307	0.7601
CATHYHIN (learn weight)	**0.4964**	**1.0618**	0.7161	**1.1283**	**1.7511**	**0.9168**

Table 3.3: Heterogeneous pointwise mutual information in NEWS (16 topics collection and 4 topics subset)

NEWS (4 topics subset)	Term-Term	Term-Person	Person-Person	Term-Location	Person-Location	Location-Location	Overall
TopK	-0.2479	0.1671	0.0716	0.0787	0.2483	0.3632	0.1317
NetClus	0.1279	0.3835	0.2909	0.3240	0.4728	0.4271	0.3575
CATHYHIN (equal weight)	**1.0471**	0.7917	0.4902	0.8506	0.6821	0.6586	0.7610
CATHYHIN (norm weight)	0.7975	0.8825	**0.5553**	0.8682	**0.8077**	0.7346	0.8023
CATHYHIN (learn weight)	0.9935	**0.9354**	0.5142	**0.9784**	0.7389	**0.7645**	**0.8434**

NEWS (16 topics)	Term-Term	Term-Person	Person-Person	Term-Location	Person-Location	Location-Location	Overall
TopK	-1.7060	-0.8663	-0.8462	-1.0238	-0.5665	-0.4578	-0.8783
NetClus	-0.3847	0.0943	0.0313	-0.1114	0.1291	0.1376	-0.0274
CATHYHIN (equal weight)	0.7804	1.0170	0.8393	0.8354	0.9467	0.6382	0.8749
CATHYHIN (norm weight)	0.8579	**1.1143**	**0.9086**	0.8530	0.9624	**0.7143**	0.9284
CATHYHIN (learn weight)	**0.9234**	1.1109	0.7966	**0.9731**	**0.9718**	0.6965	**0.9500**

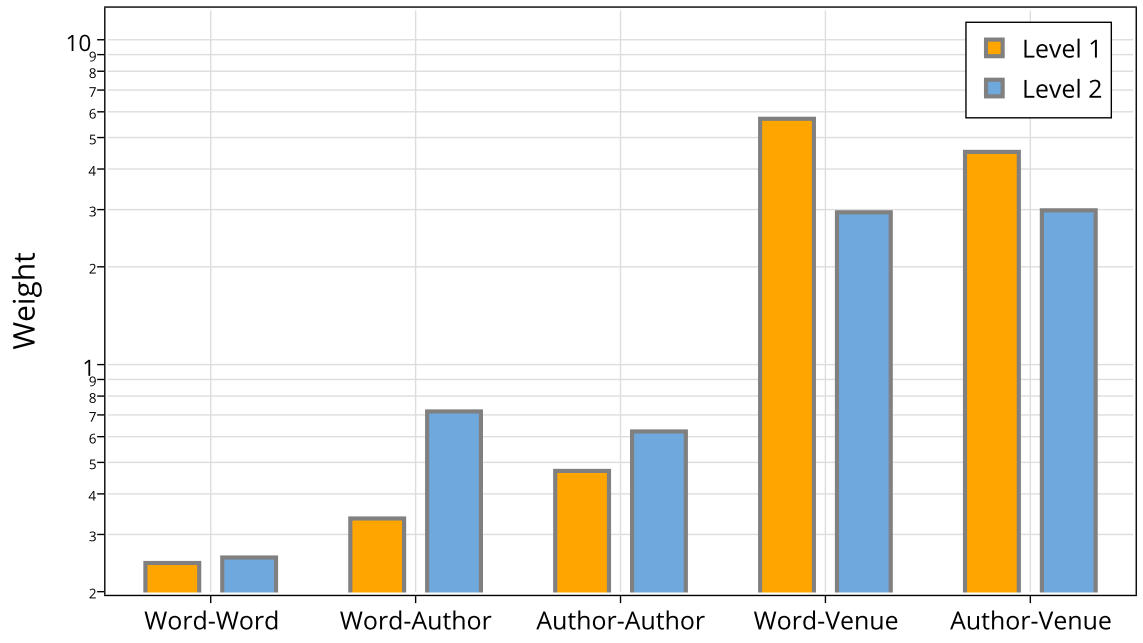


Figure 3.8: Learned link weights in DBLP

Table 3.4: # Links in our datasets

DBLP *(# Nodes)*	**Term** (6,998)	**Author** (12,886)	**Venue** (20)
Term	693,132	900,201	104,577
Author	–	156,255	99,249

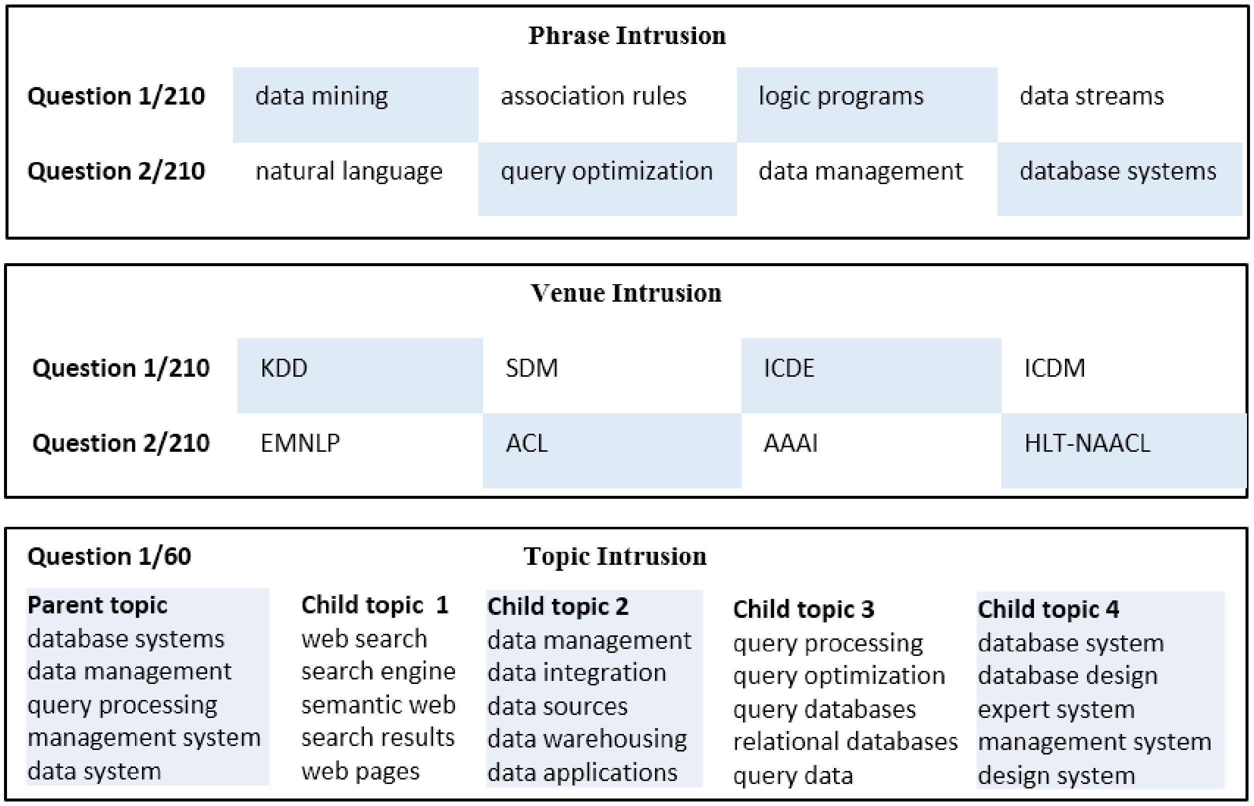
NEWS *(# Nodes)*	**Term** (13,129)	**Person** (4,555)	**Location** (3,845)
Term	686,007	386,565	506,526
Person	–	53,094	129,945
Location	–	–	85,047

Table 3.5: Results of Intruder Detection tasks (% correct intruders identified)

	DBLP				**NEWS**			
	Phrase	Venue	Author	Topic	Phrase	Location	Person	Topic
CATHYHIN	**0.83**	**0.83**	**1.0**	**1.0**	**0.65**	**0.70**	**0.80**	**0.90**
CATHYHIN$_1$	0.64	–	–	0.92	0.40	0.55	0.50	0.70
CATHY	0.72	–	–	0.92	0.58	–	–	0.65
CATHY$_1$	0.61	–	–	0.92	0.23	–	–	0.50
CATHY$_{heur\_HIN}$	–	0.78	0.94	0.92	–	0.65	0.45	0.70
NetClus$_{pattern}$	0.33	0.78	0.89	0.58	0.23	0.20	0.55	0.45
NetClus$_{pattern\_1}$	0.53	–	–	0.58	0.20	0.45	0.30	0.40
NetClus	0.19	0.78	0.83	0.83	0.15	0.35	0.25	0.45

Table 3.6: The 'information retrieval' topic, as generated by three methods

CATHYHIN	**CATHY$_{heuristic\_HIN}$**	**NetClus$_{pattern}$**
{information retrieval; web search; retrieval} / {W. Bruce Croft; Iadh Ounis; James Allen} / {SIGIR; WWW; ECIR}	{information retrieval; search engine; web search} / {Ryen W. White; C. Lee Giles; Mounia Lalmas} / {SIGIR; WWW; ECIR}	{information retrieval; statistical machine translation; conditional random fields} / {W. Bruce Croft; Zheng Chen; Chengxiang Zhai} / {ACL; SIGIR; HLT-NAACL}



Phrase Intrusion

Question 1/210	data mining	association rules	logic programs	data streams
Question 2/210	natural language	query optimization	data management	database systems

Venue Intrusion

Question 1/210	KDD	SDM	ICDE	ICDM
Question 2/210	EMNLP	ACL	AAAI	HLT-NAACL

Question 1/60 **Topic Intrusion**

Parent topic	**Child topic 1**	**Child topic 2**	**Child topic 3**	**Child topic 4**
database systems	web search	data management	query processing	database system
data management	search engine	data integration	query optimization	database design
query processing	semantic web	data sources	query databases	expert system
management system	search results	data warehousing	relational databases	management system
data system	web pages	data applications	query data	design system

Figure 3.9: Examples of intruder detection questions

Table 3.7: The ‘Egypt’ topic and the least sensible subtopic, as generated by three methods (only Phrases and Locations are shown)

CATHYHIN	**CATHY$_{\textbf{heuristic\_HIN}}$**	**NetClus$_{\textbf{pattern}}$**
{egypt; egypts; death toll; morsi} / {Egypt; Egypt Cairo; Egypt Israel; Egypt Gaza}	{egypt; egypts morsi; egypt imf loan; egypts president} / {Egypt; Cairo; Tahrir Square; Port Said}	{bill clinton; power nuclear; rate unemployment; south sudan} / {Egypt Cairo; Egypt Coptic; Israel Jerusalem; Libya Egypt}
↓	↓	↓
{death toll; egyptian; sexual harassment; egypt soccer} / {Egypt Cairo; Egypt Gaza; Egypt Israel}	{supreme leader; army general sex; court; supreme court} / {US; Sudan; Iran; Washington}	{egypts coptic pope; egypts christians; obama romney; romney campaign} / {Egypt Cairo; Egypt Coptic; Israel Jerusalem; Egypt}

Chapter 4

Topical Phrase Mining

A topic is traditionally modeled as a multinomial distribution over terms, and frequent terms related by a common theme are expected to have a large probability in a topic multinomial.

When latent topic multinomials are inferred, it is of interest to visualize these topics in order to facilitate human interpretation and exploration of the large amounts of unorganized text often found within text corpora. In addition, visualization provides a qualitative method of validating the inferred topic model [17].

While topic models have clear application in facilitating understanding, organization, and exploration in large text collections such as those found in full-text databases, difficulty in interpretation and scalability issues have hindered adoption. Several attempts have been made to address the prevalent deficiency in visualizing topics using unigrams. These methods generally attempt to infer phrases and topics simultaneously by creating complex generative mechanism. The resultant models can directly output phrases and their latent topic assignment. Two such methods are Topical N-Gram [93] and PD-LDA [54]. While it is appealing to incorporate the phrase-finding element in the topical clustering process, these methods often suffer from high-complexity, and overall demonstrate poor scalability outside small datasets.

In this section, new methods are proposed that demonstrate both scalability compared to other topical phrase mining methods and interpretability. Frequent pattern mining and statistical analysis are employed for phrase mining. We leverage the redundancy of the text in the corpus, rather than relying on linguistic analysis. Therefore, our methods do not require domain knowledge or language grammar. Also, our methods can work with informally written text such as social media messages. [1]

We introduce some notations first. The input is a corpus of D documents, where d-th document

[1]The content of this chapter is largely based on [24] and [28].

is a sequence of l_d tokens: $w_{d,i}, i = 1, \dots, l_d$. Let $L = \sum_{d=1}^{D} l_d$. For convenience we index all the unique words in this corpus using a vocabulary of V words. And $w_{d,i} = x, x \in \{1, \dots, V\}$ means that the i-th token in d-th document is the x-th word in the vocabulary. Throughout this chapter we use 'word x' to refer to the x-th word in the vocabulary.

A phrase is defined to be a sequence of words $P = \{v_i\}_{i=1}^{n}$. The length n can be any positive integer.

4.1 Criteria of good phrases and topical phrases

Our goal is to develop a mining algorithm for finding high quality candidate phrases, and design a ranking function to evaluate the quality of topical phrases. To do so, we need to understand human intuition for judging what constitutes a high quality topical phrase.

First, we notice that, it is inappropriate to discard all unigrams when approaching this task, or in fact to demonstrate a bias towards any particular phrase length. For instance, consider that the unigram 'classification' and the trigram 'support vector machines' are both high quality topical phrases for the machine learning topic in the domain of computer science. It is also not ideal to present separate ranked lists of topical phrases of each length, since when people are asked to characterize topics, they do not limit themselves to e.g. listing only bigrams, but rather provide a set of relevant phrases with no regard for phrase length. We should therefore also be able to directly compare phrases of mixed-length in a natural way. We refer to this characteristic as exhibiting the **comparability property**.

Traditional probabilistic modeling approaches, such as language models or topic models do not have the comparability property. They can model the probability of seeing an n-gram given a topic, but the probabilities of n-grams with different lengths (unigrams, bigrams, etc) are not well comparable. These approaches simply find longer n-grams to have much smaller probability than shorter ones, because the probabilities of seeing every possible unigram sum up to 1, and so do the probabilities of seeing every possible bigram, trigram, etc. However, the total number of possible n-grams grows following a power law ($\mathcal{O}(V^n)$), and ranking functions based on these traditional approaches invariably favor short n-grams. While previous work has used various heuristics to correct this bias during post-processing steps, such as using a penalization term with respect to the phrase length [81, 101], our approach is cleaner and more principled.

The key to exhibiting the comparability property is to treat each phrase as a whole unit, and design quality measure for them.

We specify two requirements of a good phrase.

- **Concordance:** In corpus linguistics, a collocation refers to the co-occurrence of tokens in such frequency that is significantly higher than what is expected due to chance. A commonly-used example of a phraseological-collocation is the example of the two candidate collocations “strong tea” and “powerful tea” [35]. One would assume that the two phrases appear in similar frequency, yet in the English language, the phrase “strong tea” is considered more correct and appears in much higher frequency. Because a collocation’s frequency deviates from what is expected, we consider them ‘interesting’ and informative. This insight motivates the necessity of analyzing our phrases probabilistically to ensure they are collocations.

- **Completeness:** If long frequent phrases satisfy the above criteria, then their subsets also satisfy these criteria. For example in the case of “mining frequent patterns”, “mining frequent” will satisfy the frequency and collocation restriction, yet is clearly a subset of a larger and more intuitive phrase. Our phrase-construction algorithm should be able to automatically determine the most appropriate size for a human-interpretable phrase.

Besides the general requirements of phrases, we wish to find high quality topical phrases that successfully represent a topic. We now present the characteristics that such a phrase should have. As a running example, consider mining and ranking phrases for topics in Computer Science.

- **Popularity:** Popularity, which may be referred to by other names such as coverage, frequency, or importance, is the most basic criteria, required by every ranking function that tackles this same problem. *Example: ‘information retrieval’ has better coverage than ‘cross-language information retrieval’ in the Information Retrieval topic.* A phrase that is not frequent in a topic should never be highly ranked as being representative of that topic, regardless of its length, or its value according to any other criteria. This further suggests that the ranking function should be designed in such a way that a topical phrase with low popularity is guaranteed to have a lower rank.

- **Purity:** A phrase is pure in a topic if it is only frequent in documents belonging to that topic and not frequent in documents within other topics. *Example: 'query processing' is more pure than 'query' in the Database topic.* Like popularity, some version of this criterion is also present in most ranking functions, though it might be referred by other names, such as 'informativeness' [81] or 'relevance' [101].

A phrase mining algorithm needs to carry out these criteria when finding high quality phrases. I present two specific mining algorithms in the next two sections. They assume the topic discovery is done with the bag-of-words assumption. For future work, one may consider using the mined phrases to regularize the topic discovery module. For example, one can induce constraints so that the tokens in the same phrase share coherent topics.

4.2 KERT: mining phrases in short, content-representative text

Like Zhao et al. [101], this section focuses on mining topical phrases from a collection of short documents. There are many cases where the full text of a document collection is not available, or is too noisy, for the desired task of topic discovery from the collection. Furthermore, we focus on documents which are information-rich, meaning that most of the document content is informative, not noisy, with the usual exception of function words. The documents may also be a mix of multiple topics, in spite of their short length. In this section we primarily evaluate the performance on two collections of scientific paper titles, which fit this criteria well. We also work with a collection of scientific paper abstracts which, while longer, may still be considered to be information-rich, and which are still significantly shorter and less noisy than the texts of full papers. While the framework could technically be applied to a collection of noisy short documents such as tweets, it would require at least a transformation of the noisy documents into information-rich documents in order to perform well, and this is out of the scope of this section.

4.2.1 Phrase quality

Concordance

A group of words should be grouped into a phrase if they co-occur significantly more frequently than the expected co-occurrence frequency given that each word in the phrase occurs independently. For example, while 'active learning' is a good phrase in the Machine Learning topics, 'learning classification' is not, since the latter two words co-occur only because both of them are popular in the topic.

We therefore compare the probability of seeing a phrase $P = \{v_1 \dots v_n\}$ and seeing the n words $v_1 \dots v_n$ independently:

$$\begin{aligned}\kappa^{con}(P) &= \log \frac{p(P)}{\prod_{v \in P} p(v)} \\ &= \log \frac{f(P)}{N} - \sum_{v \in P} \log \frac{f(v)}{N}\end{aligned} \tag{4.1}$$

where N is the total number of documents.

Completeness

A phrase P is not complete if a longer phrase P' that contains P usually co-occurs with P. For example, while 'support vector machines' is a complete phrase, 'vector machines' is not, as 'support' nearly always accompanies 'vector machines.'

We thus measure the completeness of a phrase P by examining the conditional probability of observing P' given P in a topic-t document:

$$\begin{aligned}\kappa^{com}(P) &= 1 - \max_{P' \supsetneq P} p(P'|P) \\ &= 1 - \max_{v} p(P \oplus v|P) \\ &= 1 - \frac{\max_v f(P \oplus v)}{f(P)}\end{aligned} \tag{4.2}$$

4.2.2 Topical phrase quality

We first introduce a notion of *topical frequency* of phrases.

Definition 3 (Topical frequency) *The topical frequency $f_t(P)$ of a phrase is the count of the times the phrase is attributed to topic t. For the root node o, $f_o(P) = f(P)$ is equal to the total frequency. For each topic node in the hierarchy, with C_t subtopics, $f_t(P) = \sum_{z\in[C_t]} f_{t/z}(P)$, i.e., the topical frequency is equal to the sum of the sub-topical frequencies.*

Table 4.2 illustrates an example of estimating topical frequency for phrases in a computer science topic that has 4 subtopics. The phrase 'support vector machines' is estimated to belong entirely to the Machine Learning (ML) topic with high frequency, and therefore is a candidate for a high quality phrase. However, 'social networks' is fairly evenly distributed among three topics, and is thus less likely to be a high quality phrase.

Using the learned topic model parameters in the last section, we estimate the topical frequency of a phrase $P = \{v_1 \dots v_n\}$ based on two assumptions: i) For a topic-t phrase of length n, each of the n words is generated with the distribution ϕ^t, and ii) the total number of topic-t phrases of length n is proportional to $\rho_{\pi_t,t}$.

$$f_{t/z}(P) = f_t(P)\frac{\rho_{t,z}\prod_{i=1}^{n}\phi_{v_i}^{t/z}}{\sum_{c\in C_t}\rho_{t,c}\prod_{i=1}^{n}\phi_{v_i}^{t/c}} \tag{4.3}$$

We illustrate how to use the topical frequency to quantify the two criteria about topical representativeness in Section 4.1 and combine them.

Popularity

A representative phrase for a topic should cover many documents within that topic. For example, 'information retrieval' has better coverage than 'cross-language information retrieval' in the topic of Information Retrieval. We directly quantify the popularity measure of a phrase as the probability of seeing a phrase in a random topic-t document $p(P|t)$:

$$\kappa_t^{pop}(P) = p(P|t) = \frac{f_t(p)}{N_t} \tag{4.4}$$

where N_t be the number of documents that contain at least one frequent topic-t phrase with topical frequency larger than a threshold μ.

Purity

A phrase is pure in topic t if it is only frequent in documents about topic t and not frequent in documents about other topics. For example, 'query processing' is a more pure phrase than 'query' in the Databases topic.

We measure the purity of a phrase by comparing the probability of seeing a phrase in the topic-t documents and the probability of seeing it in any other topic-t' collection ($t' = 0, 1, \ldots, k$, $t' \neq t$). If there exists a topic t' such that the probability of $p(P|t')$ is similar or even larger than $p(P|t)$, the phrase P indicates confusion about topic t and t'. The purity of a phrase compares the probability of seeing it in the topic-t collection and the maximal probability of seeing it in any mix collection:

$$\begin{aligned}\kappa_t^{pur}(P) &= \log \frac{p(P|t)}{\max_{t'} p(P|\{t, t'\})} \\ &= \log \frac{f_t(P)}{N_t} - \log \max_{t'} \frac{f_t(P) + f_{t'}(P)}{N_{\{t,t'\}}}\end{aligned} \tag{4.5}$$

where $N_{\{t,t'\}}$ is the number of documents that contain at least one frequent topic-t phrase or topic-t' phrase.

We can combine these 2 measures into a function representing the quality of a topical phrase by viewing them within an information theoretic framework. As described above, the popularity criterion is in some sense the more important, since a phrase with low popularity will be necessarily of low quality, regardless of its performance according to the other criteria. We can enforce this by representing the relationships between popularity and purity using a pointwise Kullback-Leibler(KL)-divergence metric.

Pointwise KL-divergence is a distance measure between two probabilities that takes the absolute probability into consideration, and is more robust than pointwise mutual information when the relative difference between probabilities need to be supported by sufficiently high absolute probability.[2] The product of popularity and purity, $\kappa_t^{pop}(P)\kappa_t^{pur}(P) = p(P|t)\log\frac{p(P|t)}{p(P|\{t,t^*\})}$ is equal to the pointwise KL-divergence between the probabilities of $p(P|t)$ and $p(P|\{t,t^*\})$.

Likewise, the product of popularity and concordance,
$\kappa_t^{pop}(P)\kappa_t^{con}(P)$ is equivalent to the pointwise KL-divergence between the probability of $p(P|t))$ under different independence assumptions.

Finally, we combine the two pointwise KL-divergence with a weighted summation, and implement the completeness criterion as a filtering condition to remove incomplete phrases:

$$Quality_t(P) = \begin{cases} 0 & \kappa_t^{com} \leq \gamma \\ \kappa_t^{pop}[(1-\omega)\kappa_t^{pur} + \omega\kappa^{con}](P) & \text{o.w.} \end{cases} \tag{4.6}$$

Here, $\gamma \in [0,1]$ controls how aggressively we prune incomplete phrases. $\gamma = 0$ corresponds to ignoring the completeness criterion and retaining all *closed* phrases, where no supersets have the same topical support. As γ approaches 1, more phrases will be filtered and eventually only *maximal* phrases (no supersets are sufficiently frequent) will remain. The other three criteria rank phrases that pass the completeness filter.

Although $Quality_t(P)$ is a combination of two pointwise KL-divergence metrics, the coverage criterion is a factor in both. This reflects the fact that when $p(P|t)$ is small, phrase P has low support, and thus the estimates of purity and concordance will be unreliable and their role should be limited. When $p(P|t)$ is large, phrase P has high support, and magnifies the cumulative effect (positive or negative) of the purity and concordance criteria.

The relative weights of the purity and concordance criteria are controlled by $\omega \in [0,1]$. Both measures are log ratios on comparable scales, and can thus be balanced by a weighted summation. As ω increases, we expect more topic-independent, but common phrases to be ranked higher.

[2]Note that Tomokiyo *et al.* [81] uses KL-divergence metrics to derive their ranking function as well, but they require an annotated foreground corpus and a background corpus as input. Furthermore, they consider language models only for consecutive ngrams and do not exhibit the comparability property. So their ranking function behaves quite differently from ours.

4.3 ToPMine: mining phrases in general text

KERT is simple, yet relying on the assumption that the text is short and information rich. In this section, we propose a new methodology ToPMine that works well with general text. One key difference with KERT is that it performs segmentation of each document, so that false phrases with low concordance or completeness can be better filtered.

Example 4.1 *By frequent phrase mining and context-specific statistical significance ranking, the following titles can be segmented as follows:*

Title 1. *[Mining frequent patterns] without candidate generation: a [frequent pattern] tree approach.*

Title 2. *[Frequent pattern mining] : current status and future directions.*

The tokens grouped together by [] are constrained to share the same topic assignment.

In title 1 of Example 4.1, after merging 'frequent' and 'pattern', we only need to test whether 'frequent pattern tree' is a concordant phrase in order to determine whether to keep 'frequent pattern' as a phrase in this title.

Definition 4

A phrase instance *is a sequence of contiguous tokens:* $\{w_{d,i}, ...w_{d,i+n}\}$ $n > 0$.

A partition *over d-th document is a sequence of phrases instance such that the concatenation of the phrase instances is the original document.*

In Example 4.1, we can see the importance of word proximity in phrase recognition. As such, we place a contiguity restriction on our phrases. To illustrate an induced partition upon a text segment, we can note how the concatenation of all single and multi-word phrases in Title 1 will yield an ordered sequence of tokens representing the original title.

To extract topical phrases that satisfy our desired requirements, we propose a solution that can be divided into three main parts: phrase mining, text segmentation, and topical phrase ranking. The phrase mining part finds frequent patterns; text segmentation finds high-quality phrases using the two requirements of good phrases, and feed them to topical phrase ranking.

4.3.1 Frequent phrase mining

In Algorithm 1, we present our frequent phrase mining algorithm. The task of frequent phrase mining can be defined as collecting aggregate counts for all contiguous words in a corpus that satisfy a certain minimum support threshold. We draw upon two properties for efficiently mining these frequent phrases.

1. Downward closure lemma: If phrase P is not frequent, then any super-phrase of P is guaranteed to be **not** frequent.

2. Data-antimonotonicity: If a document contains no frequent phrases of length n, the document does **not** contain frequent phrases of length $> n$.

The downward closure lemma was first introduced for mining general frequent patterns using the Apriori algorithm [2]. We can exploit this property for our case of phrases by maintaining a set of active indices. These active indices are a list of positions in a document at which a contiguous pattern of length n is frequent. In line 1 of Algorithm 1, we see the list of active indices.

In addition, we use the data-antimonotonicity property to assess if a document should be considered for further mining [38]. If the document we are considering has been deemed to contain no more phrases of a certain length, then the document is guaranteed to contain no phrases of a longer length. We can safely remove it from any further consideration. These two pruning techniques work well with the natural sparsity of phrases and provide early termination of our algorithm without searching through the prohibitively large candidate phrase space.

We take an increasing-size sliding window over the corpus to generate candidate phrases and obtain aggregate counts. At iteration k, for each document still in consideration, fixed-length candidate phrases beginning at each active index are counted using an appropriate hash-based counter. As seen in Algorithm 1 line 7, candidate phrases of length $k-1$ are pruned if they do not satisfy the minimum support threshold and their starting position is removed from the active indices. We refer to this implementation of the downward closure lemma as *position-based Apriori pruning.* As seen in Algorithm 1 lines 9 - 11, when a document contains no more active indices, it is removed from any further consideration. This second condition in addition to pruning the search for frequent phrases provides a natural termination criterion for our algorithm.

Algorithm 1: Frequent Phrase Mining

```
Input: Corpus $[D]$, min support $\mu$
Output: Frequent phrase and their frequency: $\{(P, C(P))\}$
1.1  $\mathcal{D} \leftarrow [D]$
1.2  $A_{d,1} \leftarrow \{\text{indices of all length-1 phrases} \in d\} \quad \forall d \in \mathcal{D}$
1.3  $C \leftarrow HashCounter(\text{counts of frequent length-1 phrases})$
1.4  $n \leftarrow 2$
1.5  while $\mathcal{D} \neq \emptyset$ do
1.6      for $d \in \mathcal{D}$ do
1.7          $A_{d,n} \leftarrow \{i \in A_{d,n-1} | C[\{w_{d,i}..w_{d,i+n-2}\}] \geq \mu\}$
1.8          $A_{d,n} \leftarrow A_{d,n} \setminus \{max(A_{d,n})\}$
1.9          if $A_{d,n} = \emptyset$ then
1.10             $\mathcal{D} \leftarrow \mathcal{D} \setminus \{d\}$
1.11         else
1.12             for $i \in A_{d,n}$ do
1.13                 if $i + 1 \in A_{d,n}$ then
1.14                     $P \leftarrow \{w_{d,i}..w_{d,i+n-1}\}$
1.15                     $C[P] \leftarrow C[P] + 1$
1.16     $n \leftarrow n + 1$
1.17 return $\{(P, C[P]) | C[P] \geq \mu\}$
```

The frequency criterion requires phrases to have sufficient occurrences. In general, we can set a minimum support that grows linearly with corpus size. The larger minimum support is, the more precision and the less recall is expected.

General frequent transaction pattern mining searches through an exponential number of candidates [2, 37]. When mining phrases, our contiguity requirement significantly reduces the number of candidate phrases generated. Worst case time-complexity occurs when the entire document under consideration meets the minimum support threshold. In this scenario, for a document d we generate $\mathcal{O}(l_d^2)$ (a quadratic number) candidate phrases. Although this quadratic time and space complexity seems prohibitive, several properties can be used to ensure better performance. First, separating each document into smaller segments by splitting on phrase-invariant punctuation (commas, periods, semi-colons, etc) allows us to consider constant-size chunks of text at a time. This effectively makes the overall complexity of our phrase mining algorithm linear, $\mathcal{O}(L)$, in relation to corpus size. The downward closure and data antimonotonicity pruning mechanisms serve to further reduce runtime.

4.3.2 Segmentation and phrase filtering

Traditional phrase extraction methods filter low quality phrases by applying a heuristic "importance" ranking that reflect confidence in candidate key phrases, then only keeping the top-ranked phrases [60]. Some methods employ external knowledge bases or NLP constraints to filter out phrases [56, 60].

Our candidate phrase filtering step differentiates itself from traditional phrase extraction methods by implicitly filtering phrases in our document segmentation step. By returning to the context and constructing our phrases from the bottom-up, we can use phrase-context and the partition constraints to determine which phrase-instance was most likely intended. Because a document can contain at most a linear number of phrases (the number of terms in the document) and our frequent phrase mining algorithm may generate up to a quadratic number of candidate phrases, a quadratic number of bad candidate phrases can be eliminated by enforcing the partition constraint.

The key elements of this step is our bottom-up merging process. At each iteration, our algorithm makes locally optimal decisions in merging single and multi-word phrases as guided by a statistical significance score. In the next subsection, we present an agglomerative phrase-construction algorithm then explain how the significance of a potential merging is evaluated and how this significance guides our agglomerative merging algorithm.

Phrase construction algorithm

The main novelty in our phrase mining algorithm is the way we construct our high-quality phrases by inducing a partition upon each document. We employ a bottom-up agglomerative merging that greedily merges the best possible pair of candidate phrases at each iteration. This merging constructs phrases from single and multi-word phrases while maintaining the partition requirement. Because only phrases induced by the partition are valid phrases, we have implicitly filtered out phrases that may have passed the minimum support criterion by random chance.

In Algorithm 2, we present the phrase construction algorithm. The algorithm takes as input a document and the aggregate counts obtained from the frequent phrase mining algorithm. It then iteratively merges phrase instances with the strongest association as guided by a potential merging's significance measure. The process is a bottom-up approach that induces a partition upon the original document creating a 'bag-of-phrases.'

Algorithm 2: Bottom-up Construction of Phrases from Ordered Tokens

```
Input: Counter $C$, threshold $\alpha$ of merging
Output: Partition
2.1  $H \leftarrow MaxHeap()$
2.2  Place all contiguous token pairs into $H$ with their significance score key.
2.3  while $H.size() > 1$ do
2.4      $Best \leftarrow H.getMax()$
2.5      if $Best.Sig \geq \alpha$ then
2.6          $New \leftarrow Merge(Best)$
2.7          $H.Remove(Best)$
2.8          Update significance for $New$ with its left phrase instance and right phrase instance
2.9      else
2.10         break
```

Figure 4.1 tracks the phrase construction algorithm by visualizing the agglomerative merging of phrases at each iteration with a dendogram. Operating on a paper title obtained from our DBLP titles dataset, each level of the dendogram represents a single merging. At each iteration, our algorithm selects two contiguous phrases such that their merging is of highest significance (Line 2.4) and merges them (Line 2.6–2.8). The following iteration then considers the newly merged phrase as a single unit. By considering each newly merged phrase as a single unit and assessing the significance of merging two phrases at each iteration, we successfully address the "free-rider" problem where long, unintelligible, phrases are evaluated as significant when comparing the occurrence of a phrase to the occurrence of each constituent term independently.

As all merged phrases are frequent phrases, we have fast access to the aggregate counts necessary to calculate the significance values for each potential merging. By using proper data structures, the contiguous pair with the highest significance can be selected and merged in logarithmic time, $\mathcal{O}(log(l_d))$ for each document. This complexity can once again be reduced by segmenting each document into smaller chunk by splitting on phrase-invariant punctuation. The algorithm terminates when the next merging with the highest significance does not meet a predetermined significance threshold α or when all the terms have been merged into a single phrase. This is represented by the dashed line in Figure 4.1 where there are no more candidate phrases that meet the significance threshold. Upon termination, a natural "bag-of-phrases" partition remains. While the frequent phrase mining algorithm satisfies the *frequency* requirement, the phrase construction algorithm satisfies the collocation and completeness criterion.

To statistically reason about the occurrence of phrases, we consider a null hypothesis, that the corpus is generated from a series of independent Bernoulli trials. Under this hypothesis, the presence or absence of a phrase at a specific position in the corpus is a product of a Bernoulli random variable. Under this hypothesis, the expected number of occurrences of a phrase can be interpreted as a binomial random variable. Because the number of tokens L in the corpus can be assumed to be fairly large, we can assume the skewness and that the binomial can be reasonably approximated by a normal distribution. As such the null hypothesis distribution h_0 for the random variable $f(P)$ the count of a phrase P within the corpus is:

$$h_0(f(P)) = \mathcal{N}(Lp(P), Lp(P)(1 - p(P)))) \approx \mathcal{N}(Lp(P), Lp(P)).$$

where $p(P)$ is the Bernoulli trial success probability for phrase P. The empirical probability of a phrase in the corpus can be estimated as $p(P) \approx \frac{f(P)}{L}$.

Consider a longer phrase that comprises of two phrases P_1 and P_2. The mean of its frequency under our null hypothesis of independence of the two phrases as:

$$\mu_0(f(P_1 \oplus P_2)) = Lp(P_1)p(P_2)$$

This expectation follows from treating each phrase as a constituent, functioning as a single unit in the syntax. Due to the unknown population variance and sample-size guarantees from the minimum support, we can estimate the variance of the population using sample variance: $\sigma^2_{P_1 \oplus P_2} \approx f(P_1 \oplus P_2)$, the sample phrase occurrence count.

We use a significance score to provide a quantitative measure of which two consecutive phrases form the best collocation at each merging iteration. This is measured by comparing the actual frequency with the expected occurrences under h_0.

$$sig(P_1, P_2) \approx \frac{f(P_1 \oplus P_2) - \mu_0(P_1, P_2)}{\sqrt{f(P_1 \oplus P_2)}} \tag{4.7}$$

Eq. (4.7) computes the number of standard deviations away from the expected number of occurrences under the null model. This significance score can be calculated using the aggregate counts of candidate phrases, which can be efficiently obtained from the frequent phrase-mining algorithm. This significance score can be considered a generalization of the t-statistic which has been used to identify dependent bigrams [20, 67]. By checking the h_0 of merging two contiguous sub phrases as opposed to merging each individual term in the phrase, we effectively address the 'free-rider' problem where excessively long phrases appear significant. To address the concern that the significance score relies on the naive independence assumption, we do not perform hypothesis testing to accept or reject h_0. Instead, we use the score as a robust collocation measure by which to guide our algorithm in selecting phrases to merge. A high significance indicates a high-belief that two phrases are highly associated and should be merged.

4.3.3 Topical phrase ranking

After the set of frequent phrases of mixed lengths is mined, they should be ranked with regard to the representativeness of each topic in the hierarchy, based on the popularity and purity criteria mentioned in Section 4.1.

We use the topic word distributions inferred from our model to estimate the 'topical' count $c_{i,P}(t)$ of each phrase P in each document d_i:

$$c_{i,P}(t) = c_{i,P}(\pi_t) p(t|P, \pi_t) = c_{i,P}(\pi_t) \frac{\rho_{\pi_t, \chi_t} \prod_{v \in P} \phi_{t,v}}{\sum_{z=1}^{C_{\pi_t}} \rho_{\pi_t, z} \prod_{v \in P} \phi_{\pi_t/z, v}} \tag{4.8}$$

Let the conditional probability $p(P|t)$ be the probability of "randomly choose a document and a phrase that is about topic t, the phrase is P." It can be estimated as $p(P|t) = \frac{1}{D} \sum_{i=1}^{D} \frac{c_{i,P}(t)}{\sum_{P'} c_{i,P'}(t)}$. The popularity of a phrase in a topic t can be quantified by $p(P|t)$. The purity can be measured by the log ratio of the probability $p(P|t)$ conditioned on topic t, and the probability $p(P|\pi_t)$ conditioned on its parent topic π_t: $\log \frac{p(P|t)}{p(P|\pi_t)}$.

As in KERT, a good way to combine these two factors is to use their product:

$$r_t(P) = p(P|t) \log \frac{p(P|t)}{p(P|\pi_t)} \tag{4.9}$$

which has an information-theoretic meaning: the pointwise KL-divergence between two probabilities. Finally, we use $(1 - \omega) r_t(P) + \omega p(P|t) \log sig(P)$ to rank phrases in topic t in the descending order.

4.4 Experiments

4.4.1 The impact of the four criteria

In the first set of experiments, we use the DBLP dataset - a collection of paper titles in Computer Science - to evaluate the effect of the four criteria to find topical phrases that appear to be high quality to human judges, via a user study. The titles were minimally pre-processed by removing all stopwords, resulting in 33,313 documents consisting of 18,598 unique words. We first describe the methods for comparison, and then present a sample of the phrases actually generated by these methods and encountered by participants in the user study. We then explain the details of the user study, and present quantitative results.

Ranking methods for comparison

To evaluate the performance of KERT, we implemented several variations of the function, as well as two baseline functions. The baselines come from Zhao et al. [101], who focus on topical keyphrase extraction in microblogs, but claim that their method can be used for other text collections. We implement their two best performing methods: kpRelInt* and kpRel.[3] We also construct variations of KERT with each of the four phrase quality criteria ignored in turn. We refer to these versions as $\text{KERT}_{-\text{pop}}$, $\text{KERT}_{-\text{pur}}$, $\text{KERT}_{-\text{con}}$, and $\text{KERT}_{-\text{com}}$.

These variations nicely represent the possible settings for the parameters γ and ω, which are described in Section 4.2. In KERT we set $\gamma = 0.5$ and $\omega = 0.5$. $\text{KERT}_{-\text{com}}$ sets $\gamma = 0$ to demonstrate what happens when we retain all *closed* phrases. As γ approaches 1, more phrases will be filtered but a very small number of *maximal* phrases (no supersets are frequent) will not be. $\text{KERT}_{-\text{con}}$ sets $\omega = 0$ and $\text{KERT}_{-\text{pur}}$ sets $\omega = 1$, to demonstrate the effect of ignoring concordance for the sake of maximizing purity, and ignoring purity to optimize for concordance, respectively. The minimal support for a phrase to be frequent is set $\mu = 5$.

[3]Their main ranking function kpRelInt considers the heuristics of phrase interestingness and relevance. As their interestingness measure is represented by re-Tweets, a concept that is not appropriate to our dataset, we reimplement the interestingness measure to be the relative frequency of the phrase in the dataset, and we therefore refer to our reimplementation as kpRelInt*. kpRel considers only the relevance heuristic.

Qualitative results

Table 4.3 shows the top 10 ranked topical phrases generated by each method for the topic of Machine Learning. kpRel and kpRelInt* yield very similar results, both clearly favoring unigrams. However, kpRel also ranks several phrases highly which are not very meaningful, such as 'learning classification' and 'selection learning.' Removing popularity from our ranking function yields the worst results, confirming the intuition that a high quality phrase must at minimum have good popularity. Without purity, the function favors bigrams and trigrams that all seem to be meaningful, although several high quality unigrams such as 'learning' and 'classification' no longer appear. Removing concordance, in contrast, yields meaningful unigrams but very few bigrams, and looks quite similar to the kpRelInt* baseline. Finally, without completeness, phrases such as 'support vector' and 'vector machines' are improperly highly ranked, as both are sub-phrases of the high quality trigram 'support vector machines.'

User study and quantitative results

To quantitatively measure phrase quality, we invited people to judge the generated topical phrases generated by the different methods. Since the DBLP dataset generates topics in Computer Science, we recruited 10 Computer Science graduate students - who could thus be considered to be very knowledgeable judges in this domain - for a user study.

We generated five topics from the DBLP dataset. Topic 5 is very mixed and difficult to interpret as representing just one research area. We discarded it, leaving four topics which are clearly interpretable as Machine Learning, Databases, Data Mining, and Information Retrieval. For each of the four topics, we retrieved the top 20 ranked phrases by each method. These phrases were gathered together per topic and presented in random order. Users were asked to evaluate the quality of each phrase on a 5 point Likert scale.

To measure the performance of each method given the user study results, we adapted the **nKQM@K measure** (normalized phrase quality measure for top-K phrases) from [101], which is itself a version of the *nDCG* metric from information retrieval [43]. We define nKQM@K for a method M using the top-K generated phrases:

$$nKQM@K = \frac{1}{k}\sum_{t=1}^{k}\frac{\sum_{j=1}^{K}\frac{score_{aw}(M_{t,j})}{log_2(j+1)}}{IdealScore_K}$$

Here T is the number of topics, and $M_{t,j}$ refers to the j^{th} phrase generated by method M for topic t. Unlike in [101], we have more than 2 judges, so we define $score_{aw}$ as the *agreement-weighted* average score for the $M_{t,j}$ phrase, which is the mean of the judges' score multiplied by the weighted Cohen's κ. This gives a higher value to a phrase with scores of (3,3,3) than to one with scores of (1,3,5), though the average score is identical. Finally, $IdealScore_K$ is calculated using the scores of the top K phrases out of all judged phrases.

Table 4.4 compares the performance across different methods.[4] The top performances are clearly variations of KERT with different parameter settings. As expected, KERT$_{-\text{pop}}$ exhibits the worst performance. The baselines perform slightly better, and it is interesting to note that kpRel, which is smoothed purity, performs better than kpRelInt*, and even slightly better than KERT$_{-\text{con}}$. This is because kpRelInt* adds in a measure of *overall* phrase popularity in the entire collection, which hurts rather than helps for this task. Removing completeness appears to have a very small negative effect, and we hypothesize this is because high-ranked incomplete phrases are relatively rare, though very obvious when they do occur (e.g. 'vector machines'). KERT$_{-\text{pur}}$ performs the best - which may reflect human bias towards longer phrases - with an improvement of at least 50% over the kpRelInt* baseline for all reported values of K.

Maximizing mutual information

In order to perform an objective evaluation, we use a dataset which, unlike the DBLP collection, has been labeled. In the arXiv dataset, each physics paper title is labeled by its authors as belonging to the subfield of Optics, Fluid Dynamics, Atomic Physics, Instrumentation and Detectors, or Plasma Physics. We minimally pre-processed the titles by removing all stopwords, resulting in 9,722 titles evenly sampled from the specified 5 physics subfields, and consisting of 9,648 unique words. Since the titles are labeled, we can explore which method maximizes the mutual information between

[4]Although we cannot directly evaluate which differences between the nKQM values are statistically significant, we examined the differences between the distributions of mean judge scores. We found, for example, that the preference for phrases generated by KERT$_{-\text{pur}}$ was statistically significant, whereas the difference between KERT and KERT$_{\text{-com}}$ was not.

phrase-represented topics and titles. As the collection has 5 categories, we set k=5.

For each method, we do multiple runs for various values of K (the number of top-ranked phrases per topic considered), and calculate the mutual information MI_K for that method as a function of K. To calculate MI_K, we label each of the top K phrases in every topic with the topic in which it is ranked highest. We then check each paper title to see if it contains any of these top phrases. If so, we update the number of events "seeing a topic t and category c" for $t = 1 \ldots k$, with the averaged count for all those labeled phrases contained in the title; otherwise we update the number of events "seeing a topic t and category c" for $t = 1 \ldots k$ uniformly, where c is the Primary Category label for the paper title in consideration. Finally, we compute mutual information at K:

$$MI_K = \sum_{t,c} p(t,c)\ log_2 \frac{p(t,c)}{p(t)p(c)}$$

We compare the baselines (kpRelInt* and kpRel), KERT, and variations of KERT where only popularity (KERT$_{\text{pop}}$), only purity (KERT$_{\text{pur}}$), and only coverage and purity (KERT$_{\text{pop+pur}}$) are used in the ranking function. Figure 4.2 shows MI_K for each method for a range of K.

It is clear that for MI$_{\text{K}}$, popularity is more important than purity, since KERT$_{\text{pur}}$ is by far the worst performer. Both baselines perform nearly as well as KERT$_{\text{pop}}$, and all are comfortably beaten by KERT$_{\text{pop+pur}}$ ($>$ 20% improvement for K between 100 and 600), which uses our popularity and purity measure. It is interesting to note that adding in the concordance and completeness measures yields no improvement in MI$_{\text{K}}$. However, the experiments with the DBLP dataset demonstrate that these measures - particularly concordance - are very helpful in the eyes of expert human judges. In contrast, while MI$_{\text{K}}$ is definitely improved with the addition of the purity measure, people seem to prefer that this metric not affect the phrase ranking. Although we outperform other approaches in both evaluations, these observations show interesting differences between theory-based and human-centric evaluation metrics.

4.4.2 Comparison of mining methods

We compare the following five methods:

• TNG [93] – A state-of-the art approach to n-gram topic modeling. By using additional latent variables to model bi-grams and adding word-specific multinomials, TNG can be used to construct topical phrases.

• TurboTopics [12] – A post-processing algorithm to LDA. It leverages permutation tests and a back-off n-gram language model to recursively merge same-topic terms from LDA into more understandable groupings.

• PD-LDA [54] – A hierarchical topical model that infers both phrases and topics. Using hierarchical Pitman-Yor processes, the topic is naturally shared to all constituents in a phrase.

• KERT – described in Section 4.2.

• ToPMine – described in Section 4.3.

Datasets

We use the following six datasets for evaluation purpose:

- **DBLP titles**. We collect a set of titles of recently published computer science papers. The collection has 1.9M titles, 152K unique words, and 11M tokens.
- **20Conf**. Titles of papers published in 20 conferences related to the areas of Artificial Intelligence, Databases, Data Mining, Information Retrieval, Machine Learning, and Natural Language Processing - contains 44K titles, 5.5K unique words, and 351K tokens.
- **DBLP abstracts**. Computer science abstracts containing 529K abstracts, 186K unique words, and 39M tokens.
- **TREC AP news**. News dataset(1989) containing 106K full articles, 170K unique words, and 19M tokens.
- **ACL abstracts**. ACL abstracts containing 2k abstracts, 4K unique words and 231K tokens.
- **Yelp Reviews**. Yelp reviews containing 230k Yelp reviews and 11.8M tokens.

We perform stemming on the tokens in the corpus using the porter stemming algorithm[69] to address the various forms of words (e.g. cooking, cook, cooked) and phrase sparsity. We remove English stop words for the mining and topic modeling steps. Unstemming and reinsertion of stop words are performed post phrase-mining and topical discovery.

Interpretability

We propose two user studies to evaluate the quality of mined phrases.

First, we use an *intrusion detection* task to evaluate topical separation. The intrusion detection task involves a set of questions asking humans to discover the 'intruder' object from several options (see Section 3.3.2). The results of this task evaluate how well the phrases are separated in different topics.

For each method, we sampled 20 Phrase Intrusion questions, and asked three annotators to answer each question. We report the average number of questions that is answered 'correctly' (matching the method) in Figure 4.3.

The second task is motivated by our desire to extract high-quality topical phrases and provide an interpretable visualization. This task evaluates both topical coherence on the full topical phrase list and phrase quality. We first visualize each algorithm's topics with lists of topical phrases sorted by topical frequency. For each dataset, five domain experts (computer science and linguistics graduate students) were asked to analyze each method's visualized topics and score each topical phrase list based on two qualitative properties:

- **Topical coherence:** We define topical coherence as homogeneity of a topical phrase list's thematic structure. This homogeneity is necessary for interpretability. We ask domain experts to rate the coherence of each topical phrase list on a scale of 1 *to* 10.

- **Phrase quality:** To ensure that the phrases extracted are meaningful and not just an agglomeration of words assigned to the same topic, domain experts are asked to rate the quality of phrases in each topic from 1 *to* 10.

For each expert, ratings were standardized to a z-score. We compute each algorithm's topical scores by averaging those of five experts. The results are shown in Figure 4.4 and Figure 4.5.

From Figures 4.3 and 4.4 we can tell that TopMine achieves similar performance to KERT in phrase intrusion, and demonstrates the best performance in topical coherence and phrase quality. This is mainly because KERT is designed for short text. Visual inspection suggests that many key topical unigrams are appended to common phrases, strengthening the notion of topical separation for all phrases. While this may aid KERT in phrase intrusion, we believe such practice leads to poor phrase quality, which is confirmed in Figure 4.5 as KERT demonstrates the lowest phrase-quality of the methods evaluated. A surprising occurrence is TNG and PD-LDA's poor performance in phrase intrusion. We suspect that this may be due to the many hyperparameters these complex models rely on and the difficulty in tuning them. In fact, the authors of PD-LDA make note that two of their parameters have no intuitive interpretation. Finally, Turbo Topics demonstrates above average performance on both datasets and user studies; this is likely a product of the rigorous permutation test the method employs to identify key topical phrases.

4.4.3 Scalability

To understand the run-time complexity of our phrase mining methods, we need to couple the phrase mining with certain topic modeling method in order to have a fair comparison with the other methods. We use a background LDA [24] for KERT and a phrase-constrained LDA [28] for ToPMine. For example, the phrase-constrained LDA takes the 'bag-of-phrases' output from ToPMine as constraints in LDA. By separately timing these two steps in our framework, we can empirically analyze the expected runtime of each step. Figure 4.6 demonstrates the disparity in runtime between the phrase mining and topic modeling portions of ToPMine. Displayed on a log-scale for ease of interpretation we see that the runtime of ToPMine scales linearly as we increase the number of documents (abstracts from our DBLP dataset). In addition, one can easily note that the phrase mining portion is of negligible runtime when compared to the topic modeling portion of the algorithm.

To evaluate the scalability, we test the runtime (on the same compute our framework's runtime (on the same hardware) for datasets of various sizes and domains. For some datasets, competing methods could not be evaluated due to computational complexity leading to intractable runtimes or due to large memory requirements. We have attempted to estimate the runtime based on a smaller number of iterations whenever we face computational intractability of an algorithm on a specific dataset. We used an optimized Java implementation MALLET [59] for the TNG implementation and the topic modeling portions of KERT and Turbo Topics. For PD-LDA, we used the author's original C++ code. For LDA and phrase-constrained LDA, the same JAVA implementation of phrase-constrained LDA is used (as LDA is a special case of phrase-constrained LDA). Because all these methods use Gibbs sampling to perform inference, we set the number of iterations to 1000. While we use hyperparameter optimization for our qualitative user-study tests and perplexity calculations, we do not perform hyperparameter optimization in our timed test to ensure a fair runtime evaluation. The runtime for KERT and ToPMine is the **full runtime** including both phrase mining and topic modeling.

Table 4.5 shows the runtime of each method on our datasets. As expected, complex hierarchal models such as PD-LDA display intractable runtimes outside small datasets showing several magnitudes larger runtime than all methods except Turbo Topics. Turbo Topics displays a similar runtime due to the computationally intensive permutation tests on the back-off n-gram model. These methods were only able to run on the two sampled datasets and could not be applied to the full (larger) datasets. On short documents such as titles, KERT shows great scalability to large datasets barely adding any computational costs to LDA. Yet due to KERT's pattern-mining scheme, the memory constraints and the exponential number of patterns generated make large long-text datasets intractable. ToPMine is the only method capable of running on the full DBLP abstracts dataset with runtime in the same order as LDA. Under careful observation, phrase-constrained LDA often runs in shorter time than LDA. This is because phrase-constrained LDA samples a topic once for an entire multi-word phrase, while LDA samples a topic for each word.

Tables 4.6, 4.7, 4.8 are sample results of TopMine on three relatively large datasets – DBLP abstracts, AP News articles, and Yelp reviews. ToPMine was the only method capable on running on these three large, long-text datasets. In the visualization, we present the most probable unigrams from phrase-constrained LDA as well as the most probable phrases below the unigrams. Automatic unstemming was performed as a post-processing step to visualize phrases in their most interpretable form. In many cases we see uninterpretable unigram topics that are made easier to interpret with the inclusion of topical phrases. Overall we can see that for datasets that naturally form topics such as events in the news and computer science subareas, ToPMine yields high quality topical phrases. For noisier datasets such as Yelp, we find coherent, yet lower quality topical phrases.

Figures and tables

Table 4.1: Visualization of the topic of Information Retrieval, automatically constructed by ToP-Mine from titles of computer science papers published in DBLP (20Conf dataset)

Terms	*Phrases*
search	information retrieval
web	social networks
retrieval	web search
information	search engine
based	support vector machine
model	information extraction
document	web page
query	question answering
text	text classification
social	collaborative filtering
user	topic model

Table 4.2: Example of estimating topical frequency. The topics are assumed to be inferred as machine learning, database, data mining, and information retrieval from the collection

Phrase	**ML**	**DB**	**DM**	**IR**	**Total**
support vector machines	85	0	0	0	85
query processing	0	212	27	12	251
world wide web	0	7	1	26	34
social networks	39	1	31	33	104

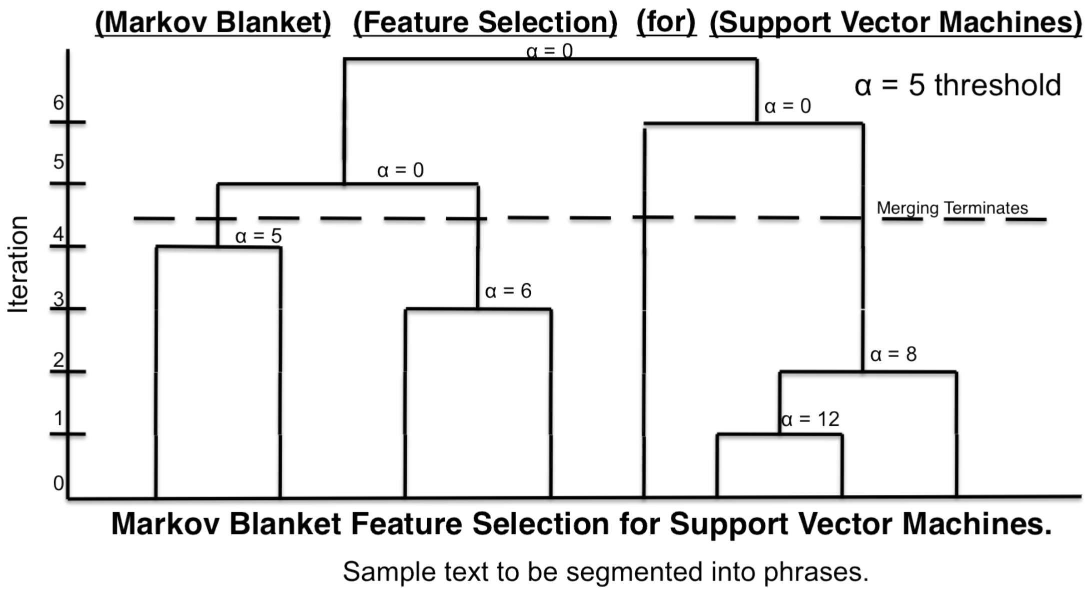

(Markov Blanket) (Feature Selection) (for) (Support Vector Machines)
α = 0
α = 0
α = 5 threshold
α = 0
Merging Terminates
α = 5
α = 6
α = 8
α = 12
Iteration
0
1
2
3
4
5
6
Markov Blanket Feature Selection for Support Vector Machines.
Sample text to be segmented into phrases.

Figure 4.1: Bottom-up construction of a 'bag-of-phrases' on computer science title taken from DBLP

Table 4.3: Top 10 ranked keyphrases in the Machine Learning topic by different methods

Method	**Top 10 Topical Keyphrases**
kpRelInt*	learning / classification / selection / models / algorithm / feature / decision / bayesian / trees / problem
kpRel	learning / classification / learning classification / selection / selection learning / feature / decision / bayesian / feature learning / trees
KERT$_{-\mathbf{cov}}$	effective / text / probabilistic / identification / mapping / task / planning / set / subspace / online
KERT$_{-\mathbf{pur}}$	support vector machines / feature selection / reinforcement learning / conditional random fields / constraint satisfaction / decision trees / dimensionality reduction / constraint satisfaction problems / matrix factorization / hidden markov models
KERT$_{-\mathbf{phr}}$	learning / classification / selection / feature / decision / bayesian / trees / problem / reinforcement learning / constraint
KERT$_{-\mathbf{com}}$	learning / support vector machines / support vector / reinforcement learning / feature selection / conditional random fields / vector machines/ classification / support machines / decision trees
KERT	learning / support vector machines / reinforcement learning / feature selection / conditional random fields / classification / decision trees / constraint satisfaction / dimensionality reduction / matrix factorization

Table 4.4: nKQM@K (methods ordered by performance)

Method	**nKQM**$_{@5}$	**nKQM**$_{@10}$	**nKQM**$_{@20}$
KERT$_{-\mathbf{pop}}$	0.2605	0.2701	0.2448
kpRelInt*	0.3521	0.3730	0.3288
KERT$_{-\mathbf{con}}$	0.3632	0.3616	0.3278
kpRel	0.3892	0.4030	0.3767
KERT$_{-\mathbf{com}}$	**0.5124**	**0.4932**	**0.4338**
KERT	**0.5198**	**0.4962**	**0.4393**
KERT$_{-\mathbf{pur}}$	**0.5832**	**0.5642**	**0.5144**

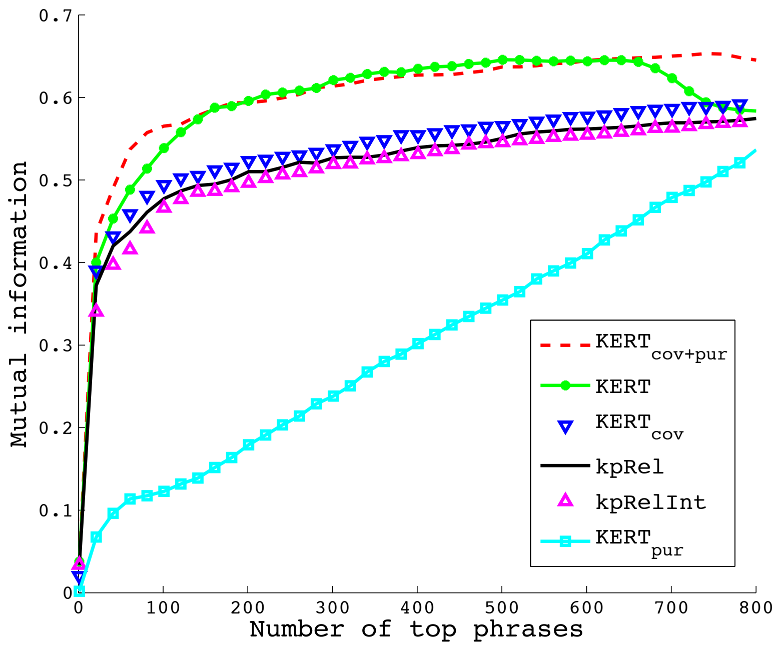


Figure 4.2: Mutual Information at K (MI_K) for various K. Methods in legend are ordered by performance, high to low

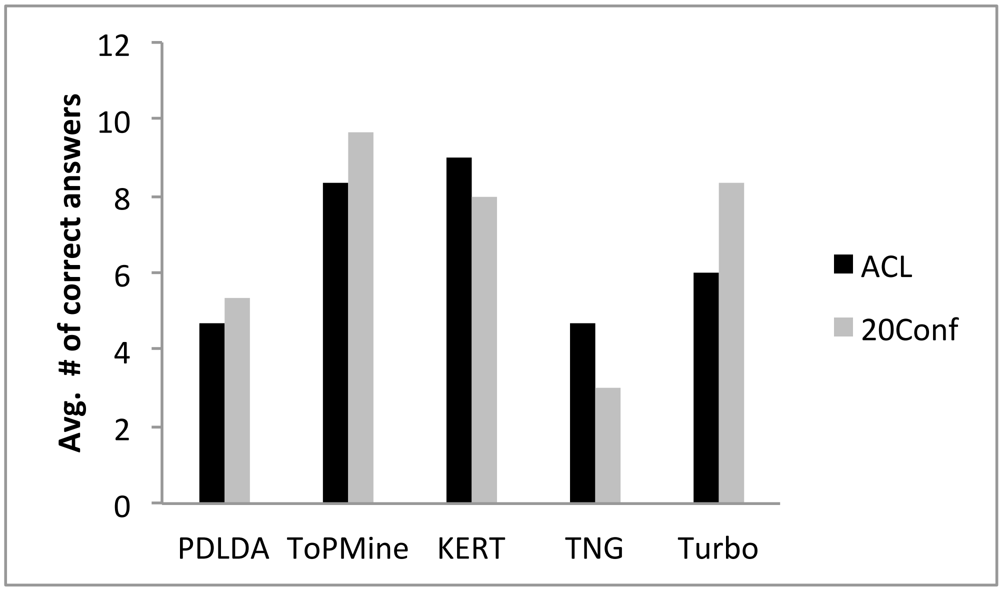


Figure 4.3: Phrase intrusion task. Test subjects were asked to identify an intruder phrase in a topic

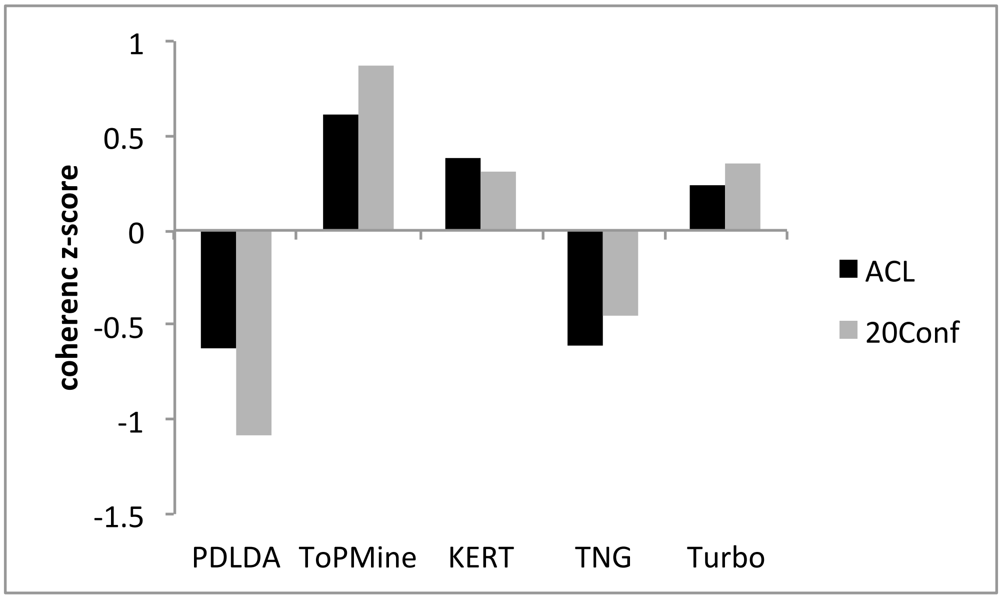


Figure 4.4: Coherence of topics. Domain experts were asked to rate the 'coherence' of each topic for each algorithm. Results were normalized into z-scores and averaged

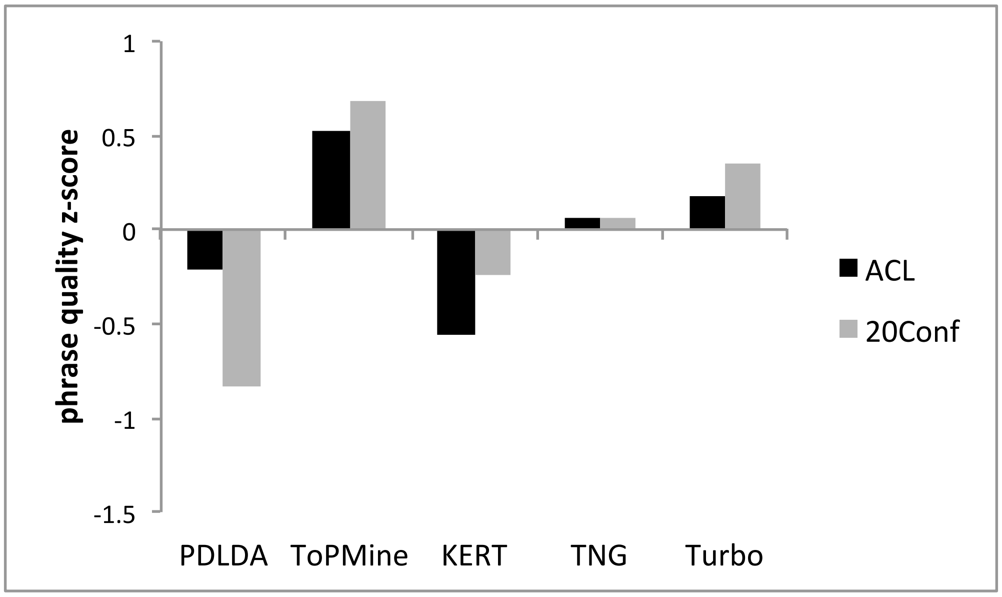


Figure 4.5: Phrase quality. Domain experts were asked to rate the quality of phrases for each topic for each algorithm. Results were normalized into z-scores and averaged

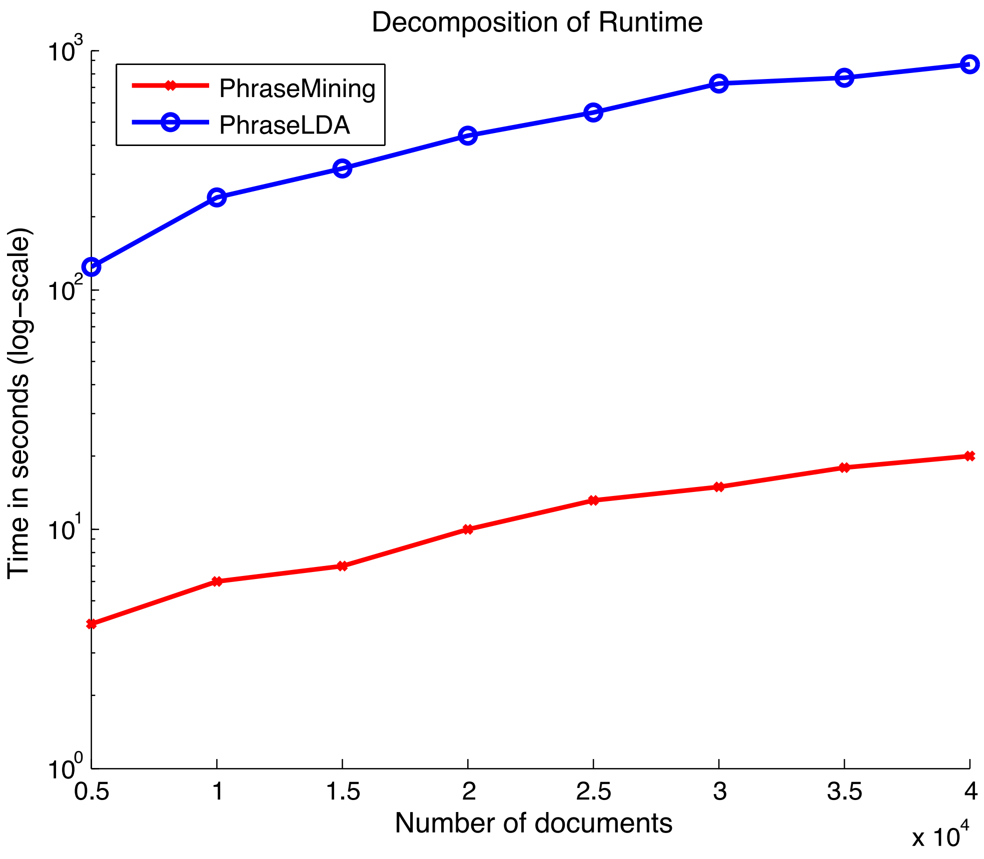


Figure 4.6: The runtime of phrase mining and phrase-constrained topic modeling. The plot above, which is displayed on a log-scale, demonstrates the speed of the phrase-mining portion. With 10 topics and 2000 Gibbs sampling iterations, the runtime of the topic modeling portion is consistently 40X the phrase mining

Table 4.5: We display the run-times of our algorithm on various datasets of different sizes from different domains. We sample 50 thousand DBLP titles and 20 thousand DBLP abstracts to provide datasets that the state-of-the art methods can perform on. For instances labeled ***, we estimate runtime by calculating the runtime for one topic and extrapolating for k topics. For instances labeled ∼ we extrapolate by calculating runtime for a tractable number of iterations and extrapolating across all iterations. For instances labeled **†**, we could not apply the algorithm to the dataset because the algorithm exceeded memory constraints (greater than 40GB) during runtime

Method	*sampled DBLP titles (k=5)*	*DBLP titles (k=30)*	*sampled DBLP abstracts*	*DBLP abstracts*
PDLDA	3.72(hrs)	∼20.44(days)	1.12(days)	∼95.9(days)
Turbo Topics	6.68(hrs)	>30(days)*	>10(days)*	>50(days)*
TNG	146(s)	5.57 (hrs)	853(s)	NA**†**
LDA	**65(s)**	3.04 (hrs)	353(s)	13.84(hours)
KERT	68(s)	3.08(hrs)	1215(s)	NA**†**
ToPMine	67(s)	**2.45(hrs)**	**340(s)**	**10.88(hrs)**

Table 4.6: Five topics from a 50-topic run of ToPMine framework on our full DBLP abstracts dataset. Overall we see coherent topics and high-quality topical phrases we interpret as search/optimization, NLP, Machine Learning, Programming Languages, and Data Mining

Topic 1	*Topic 2*	*Topic 3*	*Topic 4*	*Topic 5*
problem	word	data	programming	data
algorithm	language	method	language	patterns
optimal	text	algorithm	code	mining
solution	speech	learning	type	rules
search	system	clustering	object	set
solve	recognition	classification	implementation	event
constraints	character	based	system	time
programming	translation	features	compiler	association
heuristic	sentences	proposed	java	stream
genetic	grammar	classifier	data	large
genetic algorithm	natural language	data sets	programming language	data mining
optimization problem	speech recognition	support vector machine	source code	data sets
solve this problem	language model	learning algorithm	object oriented	data streams
optimal solution	natural language processing	machine learning	type system	association rules
evolutionary algorithm	machine translation	feature selection	data structure	data collection
local search	recognition system	paper we propose	program execution	time series
search space	context free grammars	clustering algorithm	run time	data analysis
optimization algorithm	sign language	decision tree	code generation	mining algorithms
search algorithm	recognition rate	proposed method	object oriented programming	spatio temporal
objective function	character recognition	training data	java programs	frequent itemsets

Table 4.7: Five topics from a 50-topic run of ToPMine on a large collection of AP News articles(1989). Overall we see high quality topical phrases and coherency of news topics such as environment, Christianity, Palestine/Israel conflict, Bush Administration (Senior), and health care

Topic 1	*Topic 2*	*Topic 3*	*Topic 4*	*Topic 5*
plant	church	palestinian	bush	drug
nuclear	catholic	israeli	house	aid
environmental	religious	israel	senate	health
energy	bishop	arab	year	hospital
year	pope	plo	bill	medical
waste	roman	army	president	patients
department	jewish	reported	congress	research
power	rev	west	tax	test
state	john	bank	budget	study
chemical	christian	state	committee	disease
energy department	roman catholic	gaza strip	president bush	health care
environmental protection agency	pope john paul	west bank	white house	medical center
nuclear weapons	john paul	palestine liberation organization	bush administration	united states
acid rain	catholic church	united states	house and senate	aids virus
nuclear power plant	anti semitism	arab reports	members of congress	drug abuse
hazardous waste	baptist church	prime minister	defense secretary	food and drug administration
savannah river	united states	yitzhak shamir	capital gains tax	aids patient
rocky flats	lutheran church	israel radio	pay raise	centers for disease control
nuclear power	episcopal church	occupied territories	house members	heart disease
natural gas	church members	occupied west bank	committee chairman	drug testing

Table 4.8: Five topics from a 10-topic run of our ToPMine framework on our full Yelp reviews dataset. Quality seems to be lower than the other datasets, yet one can still interpret the topics: breakfast/coffee, Asian/Chinese food, hotels, grocery stores, and Mexican food

Topic 1	*Topic 2*	*Topic 3*	*Topic 4*	*Topic 5*
coffee	food	room	store	good
ice	good	parking	shop	food
cream	place	hotel	prices	place
flavor	ordered	stay	find	burger
egg	chicken	time	place	ordered
chocolate	roll	nice	buy	fries
breakfast	sushi	place	selection	chicken
tea	restaurant	great	items	tacos
cake	dish	area	love	cheese
sweet	rice	pool	great	time
ice cream	spring rolls	parking lot	grocery store	mexican food
iced tea	food was good	front desk	great selection	chips and salsa
french toast	fried rice	spring training	farmer's market	food was good
hash browns	egg rolls	staying at the hotel	great prices	hot dog
frozen yogurt	chinese food	dog park	parking lot	rice and beans
eggs benedict	pad thai	room was clean	wal mart	sweet potato fries
peanut butter	dim sum	pool area	shopping center	pretty good
cup of coffee	thai food	great place	great place	carne asada
iced coffee	pretty good	staff is friendly	prices are reasonable	mac and cheese
scrambled eggs	lunch specials	free wifi	love this place	fish tacos

Chapter 5

Entity Topical Role Analysis

People and other entities are often characterized by the topics and themes they are working on, communicating about and involved in. The roles played by different entities in these topics are of great interest in many contexts of analysis. We may be interested in discovering the role of an author in a research community, or the contribution of a user to a social network community organized around similar interests. These types of role discovery tasks center around topical communities mined from text-attached information networks.

We are also often interested in analyzing such roles at different levels of granularity. In the real world, topical communities – communities built around shared topics – are naturally hierarchical. People participate in large communities, encompassing many interests, as well as small, focused subcommunities. Therefore, in order to analyze the various roles that an entity plays in such different contexts, we must also be able to work with topical communities and subcommunities.

In this section we study mining entity roles in hierarchical topical communities. The topical communities are discovered using the methods presented and visualized in previous two chapters. We can then discover the roles of authors who publish in these communities. For example, in the context of computer science topics, the community centered around topics on query processing and optimization may be described by the phrases {'query processing', 'query optimization',...}, while its parent community on general database topics may be described by {'query processing', 'database systems', 'concurrency control',...}. The hierarchical structure of the topical communities allows us to distinguish between, e.g., authors who publish on a diverse range of database topics, and authors who are particular experts in query processing.[1]

[1]The content of this chapter is largely based on [23].

5.1 Role of given entities

This section focuses on the type-A question in Section 1.3.1: Given a set of entities E, what are their positions?

We introduce two ways to analyze a specific entity's role in a given topic.

5.1.1 Entity specific phrase ranking

In order to specify an entity or several entities' role in a topic, we want to highlight the specific phrases which illustrate the contribution of them. We now therefore introduce an *entity specific phrase* ranking function:

$$r(P|t,E) = -p(P|t)log(\frac{p(P|t)}{p(P|t,E)}) \tag{5.1}$$

where E can be a set of entities in general. It has a nice information theoretic interpretation as the pointwise Kullback-Leibler (KL) divergence between the likelihood of seeing phrase P in the documents in topic t, and the likelihood of seeing phrase P specifically in the documents linked to entity set E, in t. Pointwise KL divergence is a distance measure between two probabilities. Therefore, $r(P|t,E)$ upranks P if its frequency in the topic in conjunction with the entity set E is higher than would be expected, based on its overall topical frequency.

$p(P|t)$ can be obtained from the topic discovery and phrase mining described in previous two chapters. We can use Bayes rule to estimate $p(P|t,E)$:

$$p(P|t,E) = \frac{p(P,E|t)}{p(E|t)} = \frac{f_t(P \cup E)\prod_{v \in P}\phi_{t,v}}{f_t(E)\sum_z \in [C_{\pi_t}]\prod_{v \in P}\phi_{\pi_t/z,v}}$$

Using only the entity specific phrase ranking does not give ideal results. Table 5.1 shows the roles of two authors, Philip S. Yu and Christos Faloutsos, in one of the subcommunities of Data Mining subtopics. Using only the contribution ranking function defined in Eq. (5.1) results in poor quality phrases such as 'fast large.' On the other hand, using only the phrase quality ranking function defined in Chapter 4 – which we refer to here as $r(P|t)$ - is also insufficient, as it only evaluates the quality of a phrase, regardless of any entity information. Therefore, we define a *combined* ranking function for a phrase P which incorporates both the relationship between the entity E and the phrase, as well as phrase quality:

$$Comb(P|t, E) = \alpha r(P|t, E) + (1 - \alpha) r(P|t) \tag{5.2}$$

The value of $\alpha \in [0, 1]$ can vary. In our experiments, we empirically set $\alpha = 0.5$. Table 5.1 illustrates that the combined ranking function yields a better list of phrases to represent the roles of the authors.

5.1.2 Distribution over subtopics

Another way to analyze the entity roles is to examine the subtopical frequency $f_{t/z}(e)$ of an entity. If the topic model generates a multinomial distribution ϕ^x for each entity type x, we can easily estimate the subtopical frequency using Bayes rule:

$$f_{t/z}(e) = f_t(e) \frac{\phi^x_{t/z,e} \rho_{t,z}}{\sum_{c \in C_t} \phi^x_{t/c,e} \rho_{t,c}} \tag{5.3}$$

Otherwise, we provide the following heuristic method to estimate entity topical frequency.

Denote the estimated number of documents in a topic t as D_t, and the set of all documents connected to E as $\mathcal{D}_E$.

For example, in the DBLP dataset, the subset $\mathcal{D}_A$ is the set of papers authored by A, and $\mathcal{D}_V$ is the set of papers published in V. Then, we need to estimate $D_{E,t}$, the number of documents attributed to E in topic t.

We must first estimate the topical frequency of every document $d_E \in \mathcal{D}_E$. In Section 4.2.2 we described how to estimate $f_t(P)$, the topical frequency of phrase P. We proceed in a similar top-down recursive fashion in order to estimate the *document* topic frequency, $f_t(d_E)$.

For each document d_E we first perform the intermediate step of calculating the *total phrase frequency* of d_E in topic t by adding up the normalized topic-t/z frequencies of all the phrases in d_E:

$$TPF_t(d_E) = \sum_{P \in d_E} \frac{f_{t/z}(P)}{\sum_{c \in C_t} f_{t/c}(P)} \tag{5.4}$$

The next step is to calculate the normalized *document frequency* of d_E in topic t:

$$f_{t/z}(d_E) = \frac{TPF_{t/z}(d_E)}{\sum_{c \in C_t} TPF_{t/c}(d_E)} f_t(d_E) \tag{5.5}$$

The topic frequency of a document is distributed among that topic's children, so that the document frequency in a given topic is the sum of the document frequencies in the topic's children, $\sum_{c \in C_t} f_{t/c}(d) = f_t(d)$. One exception is that a few documents may contain no frequent topical phrases in any subcommunities because we filter out infrequent topical phrases. For such documents we do not count their contribution to any subcommunities.

Figure 5.1 shows a hypothetical distribution of document frequency for some document. The document frequency values for every set of subcommunities sum up to the document frequency in the parent topic (where the frequency at the root is necessarily 1 for any document).

Finally, we calculate the entity topic frequency $f_t(E)$ by summing up the contributions of all the documents $d_E \in D_E$ to topic t:

$$f_t(E) = \Sigma_{d_E \in D_E} f_t(d_E) \tag{5.6}$$

Since some documents may not contribute to any of the subtopics, the entity frequency in a given topic should be equal to or larger than the sum of the entity frequencies in the topic's children, $\sum_{c \in C_t} f_{t/c}(E) \leq f_t(E)$. It is clear now that $f_t(E)$ is precisely our estimate for $D_{E,t}$.

5.1.3 Case study

As an example, Figure 5.2 and 5.3 show the roles of Christos Faloutsos and Philip S. Yu in the Data Mining topic, and its subtopic. We also show the entity frequency for each topic ($f_t(E)$), which represents the estimate for the number of papers written by that author in the topic. The sum of the entity frequencies in the subtopic do not quite add up to the entity frequency of the parent topic because, as discussed in Section 5.1.2, a document does not contribute to the child subtopics if all of its phrases have become too infrequent in them.

While both authors are prominent in the Data Mining topic, Figure 5.2 and 5.3 illustrate how their roles are contrasted in that topic, and even more strongly in the subtopic. For instance, in the third (from left) subtopic, Philip Yu contributes work on the topics of mining frequent patterns and association rules, whereas the contribution of Christos Faloutsos is more geared towards the topics of mining large datasets and large graphs.

As another example of role discovery, Figure 5.4 shows the role of the SIGIR venue in all 5 top level topics, as well as the subtopic of Machine Learning and Information Retrieval. The role of a venue in a topic is represented by those topics within the topic that are published in the venue. Thus, we can see that the Machine Learning topics that get published in SIGIR are techniques related to IR tasks such as feature selection methods that may be used for filtering, and approaches to text categorization and classification problems.

By examining the roles of different venues within a single topic, we can also gain some insight to the flavor of each venue. As an example, Table 5.2 compares the roles of three venues - SIGIR, WWW, and ECML - in the general IR topic. While both SIGIR and WWW are usually characterized as IR venues, we can clearly see that SIGIR plays a more broad role, publishing most of the topics present in the topic, whereas WWW focuses only on those topics that are directly related to the web. On the other hand, ECML is considered to be an ML venue, and its contribution to the IR topic is the publishing of papers on topics that use machine learning techniques. Note that all three venues share some high-ranked phrases, illustrating how the roles of all three venues overlap in this topic. If we were to strictly label venues, and therefore the papers they publish, as belonging exclusively to one or another topic, we would not be able to discover these interesting roles.

5.2 Entities for given roles

The role of an entity in a topical community can be interpreted as that entity's contribution to the community. For example, the role of an author is represented by the work the author does on the topics; the role of a venue is represented by the topics which get published in the venue. Therefore, a natural question to ask is which entities play the roles of top contributors to a particular topical community.

In principle, we can rank entities according to entity popularity $p(e|t)$ for a topic t. However, this entity ranking function is not able to discover entities who are more dedicated to their role in a given topic than to sibling communities. In order to take this into account, we adapt the notion of purity, as introduced in Section 4.1, to apply to entities.

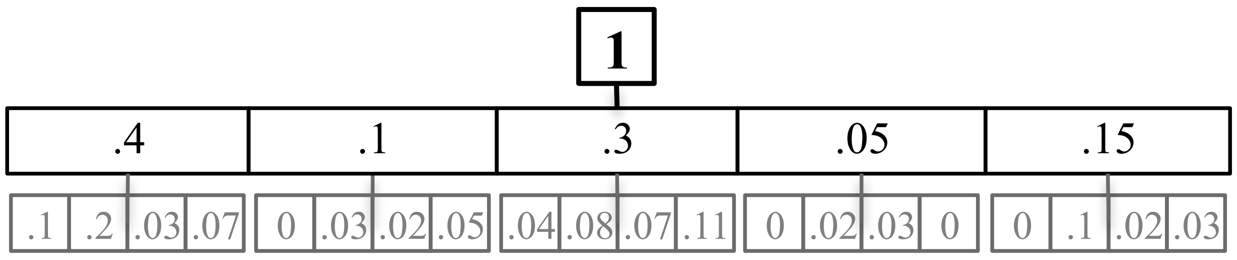


Figure 5.1: A hypothetical distribution of document frequency values for a document, in a hierarchy with 3 levels, beginning at the root

We evaluate the purity of entity e in t by comparing the probability of seeing a entity e conditioned on topic t and the contrastive probability of seeing it in a mixture of topics t and t'.

The criteria of entity purity and popularity can then be unified in an analogous way to Section 4.2. We refer to ranking entities by this value as $ERank_{Pop+Pur}$:

$$ERank_{Pop+Pur}(e,t) = p(e|t) \log \frac{p(e|t)}{p(e|t,t')}$$

Table 5.3 shows the top ranked authors in the four subtopics of Data Mining, based on $p(e|t)$ and $ERank_{Pop+Pur}$. When only coverage is used for ranking, many authors are highly ranked in all topics (e.g. Philip Yu, Jiawei Han, and Christos Faloutsos are top-5 authors in every topic). When both coverage and purity criteria are taken into account, only those authors who are significantly more dedicated to one topic are highly ranked, resulting in no overlap between topics. Some prolific authors, such as Christos Faloutsos, are no longer highly ranked anywhere, because their contributions are fairly equal among the topics. We are able to easily discover both of these roles.

Table 5.4 shows further examples of ranking authors (using $ERank_{Cov+Pur}$) within two subcommunities of the Database community. By showing the top ranked phrases for each author in a community (we discussed how these are generated in the last section) we are able to see both which authors play the most important roles, and what part of the community each author contributes to.

Figures and tables

Table 5.1: Using phrase quality, entity specific phrase ranking, and a combination of both to represent the roles of Philip S. Yu and Christos Faloutsos in a Data Mining subcommunity

Phrase Quality	**P. S. Yu (entity specific)**	**C. Faloutsos (entity specific)**	**P. S. Yu (combined)**	**C. Faloutsos (combined)**
time series	data indexing	time warping	time series	nearest neighbor
nearest neighbor	data similarity	distance	nearest neighbor	time warping
moving objects	distance	fast time	time series data	moving objects
time series data	fast large	time similarity	moving objects	nearest neighbor search
nearest neighbor queries	similarity indexing	fast large	time series mining	time series
mining time series	time series patterns	fast similarity	time series patterns	distance

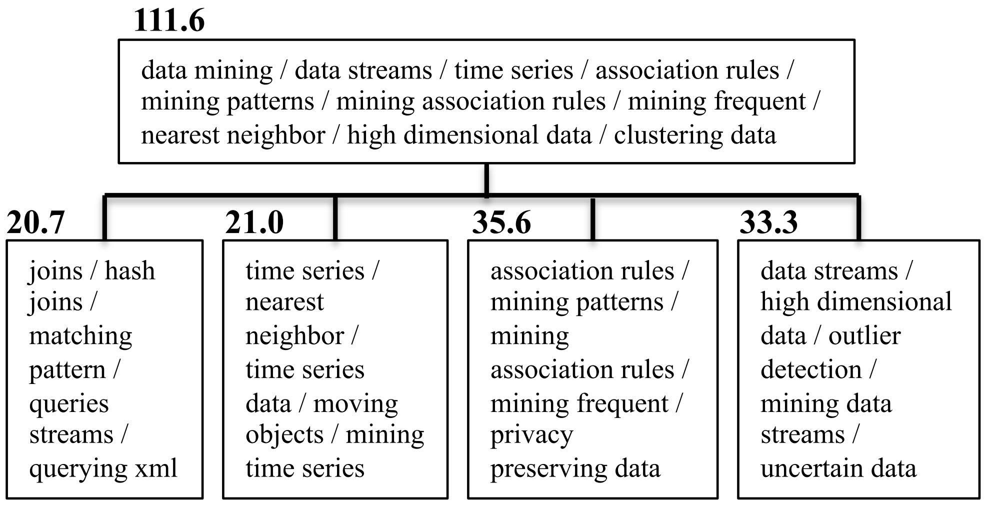
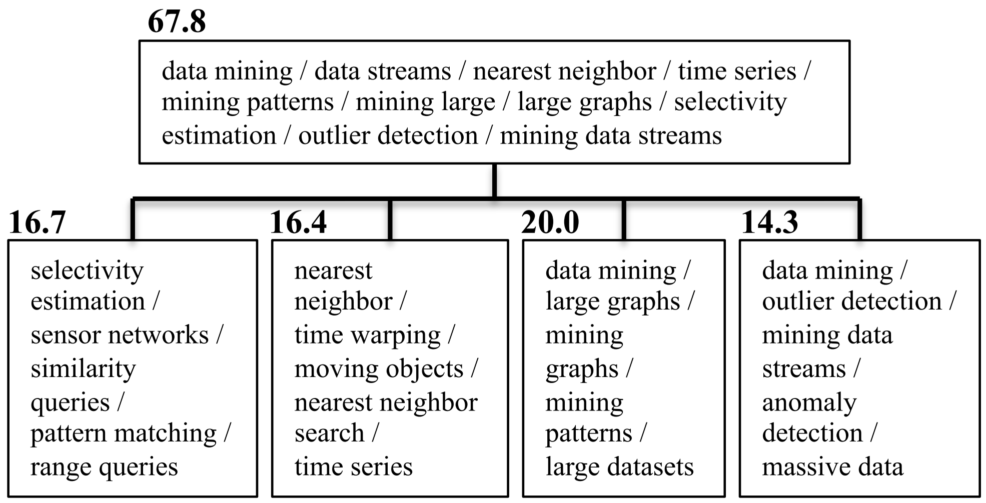

111.6
data mining / data streams / time series / association rules / mining patterns / mining association rules / mining frequent / nearest neighbor / high dimensional data / clustering data
20.7
joins / hash joins / matching pattern / queries streams / querying xml
21.0
time series / nearest neighbor / time series data / moving objects / mining time series
35.6
association rules / mining patterns / mining association rules / mining frequent / privacy preserving data
33.3
data streams / high dimensional data / outlier detection / mining data streams / uncertain data

Figure 5.2: The roles of Philip S. Yu in Data Mining



67.8
data mining / data streams / nearest neighbor / time series / mining patterns / mining large / large graphs / selectivity estimation / outlier detection / mining data streams
16.7
selectivity estimation / sensor networks / similarity queries / pattern matching / range queries
16.4
nearest neighbor / time warping / moving objects / nearest neighbor search / time series
20.0
data mining / large graphs / mining graphs / mining patterns / large datasets
14.3
data mining / outlier detection / mining data streams / anomaly detection / massive data

Figure 5.3: The roles of Christos Faloutsos in Data Mining

Table 5.2: The roles of three venues - SIGIR, WWW, and ECML - in the general Information Retrieval topic

SIGIR	**WWW**	**ECML**
information retrieval	web search	word sense disambiguation
question answering	semantic web	world wide web
web search	search engine	information extraction
natural language	question answering	semantic role labeling
document retrieval	web pages	knowledge discovery
relevance feedback	world wide web	query expansion
query expansion	web services	machine translation

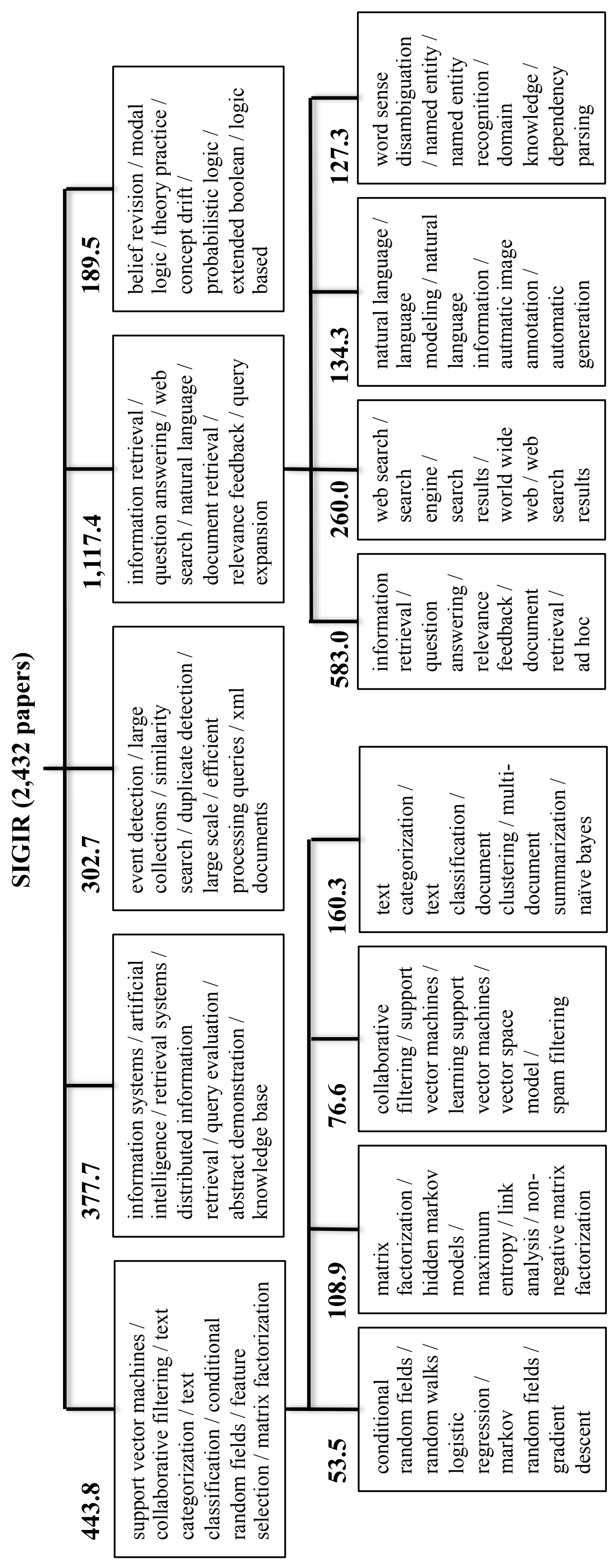

SIGIR (2,432 papers)
443.8
support vector machines / collaborative filtering / text categorization / text classification / conditional random fields / feature selection / matrix factorization
377.7
information systems / artificial intelligence / retrieval systems / distributed information retrieval / query evaluation / abstract demonstration / knowledge base
302.7
event detection / large collections / similarity search / duplicate detection / large scale / efficient processing queries / xml documents
1,117.4
information retrieval / question answering / web search / natural language / document retrieval / relevance feedback / query expansion
189.5
belief revision / modal logic / theory practice / concept drift / probabilistic logic / extended boolean / logic based
53.5
conditional random fields / random walks / logistic regression / markov random fields / gradient descent
108.9
matrix factorization / hidden markov models / maximum entropy / link analysis / non-negative matrix factorization
76.6
collaborative filtering / support vector machines / learning support vector machines / vector space model / spam filtering
160.3
text categorization / text classification / document clustering / multi-document summarization / naïve bayes
583.0
information retrieval / question answering / relevance feedback / document retrieval / ad hoc
260.0
web search / search engine / search results / world wide web / web search results
134.3
natural language / language modeling / natural language information / autmatic image annotation / automatic generation
127.3
word sense disambiguation / named entity / named entity recognition / domain knowledge / dependency parsing

Figure 5.4: The role of the venue SIGIR in several topics and subtopics. The estimated number of papers published in SIGIR within each topic is also shown

Table 5.3: Top ranked authors in the four subtopics of Data Mining, based on popularity only, and popularity + purity

{*sensor networks, selectivity estimation, large databases, pattern matching,spatio-temporal moving objects, large collections*}	{*time series, nearest neighbor, moving objects, time series data, nearest neighbor queries*}	{*association rules, large scale, mining association rules, privacy preserving, frequent itemsets*}	{*high dimensional, data streams, data mining, high dimensional data, outlier detection*}
divesh srivasta	eamonn j. keogh	jiawei han	philip s. yu
nick koudas	philip s. yu	philip s. yu	jiawei han
jiawei han	christos faloutsos	jian pei	charu c. aggarwal
philip s. yu	hans-peter kriegel	christos faloutsos	jian pei
christos faloutsos	jiawei han	ke wang	christos faloutsos
divesh srivasta	eamonn j. keogh	jiawei han	charu c. aggarwal
surat chaudhiri	jessica lin	ke wang	graham cormode
nick koudas	michail vlachos	xifeng yan	s. muthukrishnan
jeffrey f. naughton	michael j. passani	bing liu	philip s. yu
yannis papakonstantinou	matthias renz	mohammed j. zaki	xiaolei li

Table 5.4: The top ranked authors (using $ERank_{Pop+Pur}$) in two subtopics of the Database topic, along with each author's top ranked phrases in each topic. Each subtopic is represented by its top-ranked phrases, shown in the first row of each table

{*query processing / query optimization / deductive databases / materialized views / microsoft sql server / relational databases*}	
elke a. rundensteiner	query processing / query optimization / materialized views / stream processing / object-oriented databases
hamid pirahesh	query processing / query optimization / materialized views / relational data / relational xml
surajit chaudhuri	query optimization / relational databases / microsoft sql server / materialized views / relational data
jeffrey f. naughton	materialized views / xml query / query processing / relational xml / maintenance view
per-åke larson	materialized views / microsoft sql server / query optimization / materialized maintenance views / relational data
vivek r. narasayya	microsoft sql server / materialized views / relational databases / query management / sql data
serge abiteboul	materialized views / xml data / schemas / query evaluation / materialized maintenance views

{*concurrency control / database systems / main memory / load shedding / database concurrency control / load balancing*}	
avi silberschatz	concurrency control / main memory / locking / database systems / transaction management
david b. lomet	recovery / systems recovery / b-trees / transactions recovery / performance access
henry k. korth	concurrency control / database systems / main memory / protocol / transaction systems
bharat k. bhargava	concurrency control / distributed systems / distributed database / recovery / distributed database systems
c. mohan	concurrency control / recovery / locking / data systems / transaction systems
ahmed k. elmagarmid	database systems / concurrency control / distributed database / distributed systems / access control
nancy lynch	concurrency control / locking / nested transactions / control transactions / concurrency transactions

Chapter 6

Mining Hierarchical Relations

In this chapter, we discuss an alternative way of positioning entities, by mining latent relations among entities. Especially, we focus on the problem of mining hierarchical relations due to its particular interest to many applications reviewed in Section 2.3. However, our methodology can be applied to general relation mining problems as well.

The goal is to find a hierarchy, in which each node represents an object, and each edge represents a certain relation between the two objects, such as advisor-advisee. The role of each object is determined by its position in this hierarchy.

Formalization. Given a set of objects $\mathcal{V} = \{v_1, \ldots, v_n\}$ and their directed links $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$, a relation $R \subset \mathcal{E}$ is a **hierarchical relation** if (1) for every object $u \in \mathcal{V}$, there exists exactly one object $v \in \mathcal{V}$ such that $(u, v) \in R$; and (2) there does not exist a cycle along R, *i.e.*, no sequence $(u_1, \ldots, u_t), t > 1$, such that $u_1 R\, u_2, \ldots, u_{t-1} R\, u_t, u_t R\, u_1$. In the relation instance $v_i R\, v_j$, we name v_i as a *child* and v_j a *parent*. For convenience we denote (v_i, v_j) to be e_{ij}. We define the out-neighbors of one node as $Y_i = \{v_j | (v_i, v_j) \in \mathcal{E}\}$.

In general, we want to predict for every pair of directly linked objects $(v_i, v_j) \in \mathcal{E}$, whether the statement '$(v_i, v_j) \in R$' is true.

Proposed approach:

Step 1. Identify a partial order along the links in $\mathcal{E}$, and construt a directed acyclic graph (DAG) of candidate relations.

Step 2. Use a probabilistic graphical model to model the dependency of all the relations.

Step 3. If training data is provided, learn the model parameters by maximum a posterior (MAP) inference.

Step 4. Maximize the joint likelihood and predict true relations.

This framework is applicable to various domains. It is generic and does not restrain to text data or linked data. It can work with text-only data, link-only data or text plus links. The following two sections illustrate the proposed approach in unsupervised and supervised settings. Step 3 only exists in the supervised setting.[1]

6.1 Unsupervised setting

In the unsupervised setting, we have only a few number of features and constraints as the expectation of the roles. They are either equally important for relation discovery, or have unequal but known importance.

We take a case study of advisor-advisee relationship. In this section, we take a computer science bibliographic network as an example, to analyze the roles of authors and to discover the likely advisor-advisee relationships. To clearly illustrate the problem, Figure 6.2 gives an example of advisor-advisee relationship mining. The left figure shows the input: an temporal collaboration network, which consists of authors, papers, and paper-author relationships. The middle figure shows the output of our analysis: an author network with solid arrow indicating the advising relationship, and dotted arrow suggesting potential but less probable relationship. For example, the arrow from Bob to Ada indicates that Ada is identified as the advisor of Bob. The triple on the edge, i.e., $(0.8, [1999, 2000])$, represents Ada has the probability of 80% to be the advisor of Bob from 1999 to 2000. The right figure gives an example of visualized chronological hierarchies. The parent-child relation in the tree corresponds to the advisor-advisee relationship.

The following subsections describe our solutions.

6.1.1 Notations

In this subsection, we define notations used throughout this section.

In general, our study takes as input a time-dependent collaboration network $H = \{(\mathcal{V} = \mathcal{V}^p \cup \mathcal{V}^a, \mathcal{E})\}$, where $\mathcal{V}^p = \{p_1, \ldots, p_{n_p}\}$ is the set of publications, with p_i published in time t_i, $\mathcal{V}^a = \{a_1, \ldots, a_{n_a}\}$ is the set of authors, and $\mathcal{E}$ is the set of edges. Each edge $e_{ij} \in \mathcal{E}$ associates the paper p_i and the author a_j, meaning a_j is one author of p_i.

The original heterogeneous network can be transformed into a homogeneous network containing

[1]The content of this chapter is largely based on [83, 92, 87].

only authors. Let $G = (\mathcal{V}^a \cup \{a_0\}, \mathcal{E}', \{\mathbf{py}_{ij}\}_{e_{ij}\in\mathcal{E}'}, \{\mathbf{pn}_{ij}\}_{e_{ij}\in\mathcal{E}'})$, where a_0 is a virtual author, which will be the root of an advising tree. Each edge $e'_{ij} = (i,j) \in \mathcal{E}$ connects authors a_i and a_j if and only if they have publication together, and there are two vectors associated with the edge, Pub_Year_vector $\mathbf{py}_{ij}$ and Pub_Num_vector $\mathbf{pn}_{ij}$. They are of equivalent length, indicating the year they have publications and the number of coauthored papers they have at that time. For example, $\mathbf{py}_{ij} = (1999, 2000, 2001), \mathbf{pn}_{ij} = (2,3,4)$ indicates that author a_i and a_j have coauthored 2, 3 and 4 papers in 1999, 2000, and 2001 respectively. Similarly, we associate with each author two vectors $\mathbf{py_i}$ and $\mathbf{pn_i}$ to respectively represent the number of papers and the corresponding published year by author a_i. The two vectors $\mathbf{py_i}$ and $\mathbf{pn_i}$ can be derived from $\mathbf{py_{ij}}$ and $\mathbf{pn_{ij}}$.

We denote the author a_i's advisor as a_{y_i}, where y_i is an introduced hidden variable. If a_i's advisor is a_j, we use $[st_{ij}, ed_{ij}]$ to represent the time interval the advising relation lasts. For brevity we denote $st_i = st_{iy_i}$ and $ed_i = ed_{iy_i}$. If a_i is not advised by anybody in the database, we let $y_i = 0$ to direct a_i's advisor to a virtual node a_0.

In this setting, to find the advisor-advisee relationship, we need not only to decide the value of the hidden variable y_i for each author a_i, but also to estimate the start and the end years st_{iy_i}, ed_{iy_i}. In reality, this problem is more complicated: (i) one could have multiple advisors like master advisors, PhD co-advisors, post-doctorial advisors; (ii) some mentors from industry behave similarly as academic advisors if only judged by the collaboration history; and (iii) one's advisor could be missing in the data set. Therefore, instead of using a boolean model, we adopt a probabilistic model to rank the likelihood of potential advisor(s) for each author. Formally, we denote r_{ij} as the probability of a_j being the advisor of a_i. To reduce the number of authors being ranked, it is beneficial to keep only those potential pairs of advisor-advisee. We construct a subgraph $G' \subset G$ by removing some edges from G and make the remaining edges directed from advisee to potential advisor. Thus $G' = (\mathcal{V}^a \cup \{a_0\}, \mathcal{E}'_s)$ and $\mathcal{E}'_s \subset \mathcal{E}'$. Later we will show that it is possible to extract a directed acyclic graph (DAG) G' from G. In G', the index set of potential advisors of a given author a_i is denoted $Y_i = \{j | e_{ij} \in \mathcal{E}'_s\}$, e.g., $Y_3 = \{0, 1\}$. Correspondingly, the index set of potential advisees is denoted $Y_i^{-1} = \{j | e_{ji} \in \mathcal{E}'_s\}$.

Then the task becomes finding r_{ij}, st_{ij}, ed_{ij} for every possible advising pair $(i,j) \in \mathcal{E}'_s$. So the output is the DAG $H' = (\mathcal{V}^a \cup \{a_0\}, \mathcal{E}'_s, \{(r_{ij}, st_{ij}, ed_{ij})\}_{(i,j)\in\mathcal{E}'_s})$. After the chronological DAG H' is calculated, the ranking score can be used to predict whether there is an advisor-advisee relationship between every pair of coauthors (a_i, a_j). A simple way to predict is to fetch top k potential advisors of a_i and check whether a_j is one of them while $r_{ij} > r_{i0}$ or $r_{ij} > \theta$, where θ is a threshold such as 0.5. We use $P@(k, \theta)$ to denote this method. It is predictable that large k and large θ leads to better recall and worse precision. How to choose k and θ could be a tricky problem. So we allow the input contains some training data so as to determine the parameters. If no training data is provided, we can simply use some empirical values, such as the third quartile of all the ranking scores.

6.1.2 Assumptions and framework

Commonsense knowledge is needed for recognizing interesting semantic relationships. Here we make a few general assumptions based on the commonsense knowledge about advisor-advisee relationships.

Assumption 6.1 $\forall 1 \le x \le n_a, ed_{y_x} < st_x < ed_x$

This formula reflects the following fact for general consideration of advising relationship. At each time t during the publication history of a node x, x is either being advised or not being advised. Once x starts to advise another node, it will never be advised again. x cannot advise y at the year t_1 if x is advised by any node p at the year t_1. If x advises y, the time y is advised by x is a continuous interval from t_1 to t_2, $t_1 < t_2$. As a result of Assumption 6.1, we need to infer the advisors of all the nodes in the network together, rather than consider them separately. In Section 6.1.4, we will use this assumption in our model.

Assumption 6.2 $\forall 1 \le x \le n_a, py^1_{y_x} < py^1_x$

That means for a given pair of advisor and advisee, the advisor always has a longer publication history than the advisee. py^1_x represents the first component of vector $\mathbf{py}_x$. Assumption 6.2 determines that all the authors in the network have a strict order defined by the possible advising relationship. Due to the order, the candidate graph G' is assured to be a DAG. We will use this assumption in the filtering process in Section 6.1.3.

Additional assumptions about the correlation between the potential relationship and the publication history will be discussed in Section 6.1.3. Now we propose a two-stage framework solution for the advisor-advisee relationship mining problem. In stage 1, we preprocess the heterogeneous collaboration network to generate the candidate graph G'. This includes the transformation from H to a homogeneous network G, the construction from G to G', and the estimate of the local likelihood on each edge of G'. In stage 2, these potential relations are further modeled with a probabilistic model. Local likelihood and time constraints are combined in the global joint probability of all the hidden variables. The joint probability is maximized and the ranking score of all the potential relations is computed together. The construction of H' is finished in this stage.

6.1.3 Stage 1: preprocessing

The purpose of preprocessing is to generate the candidate graph G' and reduce the search space while keeping the real advisor not excluded from the candidate pool in most cases. First, we need to generate according to the collaboration information a homogeneous author network G by processing the papers in the network one by one. For each paper $p_i \in \mathcal{V}^p$, we construct an edge between every pair of its authors and update the vectors $\mathbf{py}$ and $\mathbf{pn}$. The complexity of this process is $\mathcal{O}(\sum_{p_i \in \mathcal{V}^p} \delta_i^2)$, where δ_i is the degree of p_i in H.

Then a filtering process is performed to remove unlikely relations of advisor-advisee. For each edge e_{ij} on G, a_i and a_j has collaboration. To decide whether a_j is a_i's potential advisor, the following conditions are checked. First, Assumption 6.2 is checked. Only if a_j started to publish earlier than a_i, the possibility is considered. Second, some heuristic rules are applied, which are based on the prior intuitive knowledge about advisor-advisee relations. Many rules are reasonable but for each there is counter example in real world. It is unknown how well they work before the results are tested. Thus we list the rules here and will test them in the experiment part.

First, we introduce two measures for the coauthored publications between any pair of collaborators, *kulc* (*i.e.*, Kulczinski measure [96] and IR (*i.e.*, imbalance ratio). They are defined as

$$kulc_{ij}^{t} = \frac{\sum_{py_{ij}^k \le t} pn_{ij}^k}{2} \left(\frac{1}{\sum_{py_i^k \le t} pn_i^k} + \frac{1}{\sum_{py_j^k \le t} pn_j^k} \right) \tag{6.1}$$

$$IR_{ij}^t = \frac{\sum_{py_j^k \le t} pn_j^k - \sum_{py_i^k \le t} pn_i^k}{\sum_{py_i^k \le t} pn_i^k + \sum_{py_j^k \le t} pn_j^k - \sum_{py_{ij}^k \le t} pn_{ij}^k} \tag{6.2}$$

The Kulczynski measure reflects the correlation of the two authors' publications. [96] shows that there usually exists high correlation between the total publications of advisors and advisee. Here we further incorporate the time factor, to calculate the measure year by year, and check whether there is an increase in the sequence $\{kulc_{ij}^t\}_t$. For IR, we calculate the sequences in the same way. IR [96] is used to measure the imbalance of the occurrence of a_j given a_i and the occurrence of a_i given a_j. The intuition is that the advisor has more publications than the advisee during the advising time. Then we have the following rule.

Author a_j is not considered to be a_i's advisor if one of the following conditions holds:

$R1$: $IR_{ij}^t < 0$ in the sequence $\{IR_{ij}^t\}_t$ during the collaboration period of a_i and a_j,

$R2$: there is no increase in the sequence $\{kulc_{ij}^t\}_t$ during the collaboration period,

$R3$: the collaboration period of a_i and a_j lasts only for one year,

$R4$: $py_j^1 + 2 > py_{ij}^1$,

When the pair of authors passes the test of selected rules from them, we construct a directed edge from a_i to a_j in G'. In addition, we estimate the starting time and ending time of the advising, as well as the local likelihood of a_j being a_i's advisor l_{ij}. For the estimation we also have various methods. The starting time st_{ij} is estimated as the time they started to collaborate, while the ending time ed_{ij} can be estimated as either the time point when the Kulczynski measure starts to decrease, or the year making the largest difference between the Kulczynski measure before and after it. We refer to the two methods as YEAR1 and YEAR2. And we refer to YEAR as taking the earlier time of the two years estimated by them. After estimating st_{ij} and ed_{ij}, we calculate the average of Kulczynski and IR measure during that period, and use 1)Kulczynski ; 2)IR; 3)the average of the two as three different definitions of the local likelihood. The last definition is formally

$$l_{ij} = \frac{\sum_{st_{ij} \le t \le ed_{ij}} (kulc_{ij}^t + IR_{ij}^t)}{2(ed_{ij} - st_{ij} + 1)} \tag{6.3}$$

And the other two are similar. The complexity of processing each edge is $\mathcal{O}(\Delta)$, if we assume the oldest paper and the newest one differs Δ in their publication time. The total complexity to transform G to G' is $\mathcal{O}(M\Delta)$, where M is the number of edges in G.

6.1.4 Stage 2: TPFG model

From the candidate graph G' we know the potential advisors of each author and the likelihood based on local information. By modeling the network as a whole, we can incorporate both structure information and temporal constraint and better analyze the relationship among individual links. Now we define the TPFG model.

For each node a_i, there are three variables to decide: y_i, st_i, and ed_i. Suppose we have already had a local feature function $g(y_i, st_i, ed_i)$ defined on the three variables of any given node. To model the joint probability of all the variables in the network, we define it as the product of all local feature functions.

$$p(\{y_i, st_i, ed_i\}_{a_i \in \mathcal{V}^a}) = \frac{1}{Z} \prod_{a_i \in \mathcal{V}^a} g(y_i, st_i, ed_i) \tag{6.4}$$

with

$$\forall a_i \in \mathcal{V}^a, ed_{y_i} < st_i < ed_i \tag{6.5}$$

where $\frac{1}{Z}$ is the normalizing factor of the joint probability

Eq. (6.5) is the constraint according to Assumption 6.1. To find the most probable values of all the hidden variables, we need to maximize the joint probability of all of them. To estimate the approximate size of the entire search space, assume each author has C candidates and the advising time can vary in a range of Δ, then the combination of all the variables has exponential size $(C\Delta^2)^{n_a}$. It is intractable to do exhaustive search. We make the first simplification as follows. Suppose a_i and his advisor y_i are given. Instead of letting st_i and ed_i vary, we fix them by optimizing local function $g(y_i, st_i, ed_i)$, *i.e.*,

$$\{st_i, ed_i\} = arg \max_{st_i < ed_i} g(y_i, st_i, ed_i) \tag{6.6}$$

In this way, st_i and ed_i are tied to the value of y_i. Once y_i is decided, they are derived correspondingly. We can pre-compute the best advising time as st_{ij} and ed_{ij} for each $y_i = j$. Now only $\{y_i\}$ are variables to optimize). If we embed the constraint Eq. (6.5) into the feature function, the objective function becomes

$$p(y_1, \dots, y_{n_a}) = \frac{1}{Z} \prod_{i=1}^{n_a} f_i(y_i | \{y_x | x \in Y_i^{-1}\}) \tag{6.7}$$

with

$$f_i(y_i = j | \{y_x | x \in Y_i^{-1}\}) = g(y_i, st_{ij}, ed_{ij}) \prod_{x \in Y_i^{-1}} I(y_x \neq i \vee ed_{ij} < st_{xi}) \tag{6.8}$$

where

$$I(y_x \neq i \vee ed_{ij} < st_{xi}) = \begin{cases} 1 & y_x \neq i \vee ed_{ij} < st_{xi} \\ 0 & y_x = i \wedge ed_{ij} >= st_{xi} \end{cases} \tag{6.9}$$

is the identity function. If any author a_x is advised by a_i and their advising time conflict, the function takes 0; otherwise it takes 1. In this way the time constraints Eq. (6.5) for all hidden variables are decomposed to many local identity function. Now we only need to optimize Eq. (6.7). Furthermore, to obtain the rank score of each advising relationship, *e.g.*, a_j advise a_i (shortly $a_j \rightarrow a_i$), we can compute the conditional maximal probability

$$r_{ij} = \max p(y_1, \dots, y_{n_a} | y_i = j) \tag{6.10}$$

This simplification assures that for each configuration of $\{y_i\}$, the solution achieves either 0 or the conditional optimum given that configuration. The search space size now becomes C^{n_a}, reduced but still exponential. Since we have decomposed the dependency of the variables, we can use a factor graph model to accomplish efficient computation.

Figure 6.3 shows a simple TPFG corresponding to the example we have been using so far. The graph is composed of two kinds of nodes: variable nodes and function nodes. Variable nodes map to the hidden variables $\{y_i\}_{i=0}^{n_a}$. Each variable node corresponds to a function node $f_i(y_i | \{y_x | x \in Y_i^{-1}\})$. All of the edges are of one kind, connecting a variable node with a function node. There is an edge between one variable node y_x and a function node $f_i(.)$ if and only if $f_i(.)$ depends on y_x. In our case, it is equivalent with $x = i$ or $x \in Y_i^{-1}$ (a.k.a. $i \in Y_x$). The factor graph reflects the dependency of the variables. A set of variables are correlated if they are neighbors of the same function node, *e.g.*, y_1, y_2, y_3 with $f_1(y_1, y_2, y_3)$. We can see that two hidden variables are correlated iff their corresponding author nodes are linked by an edge on the candidate graph H', which means there is a potential advising relationship between them. And once a variable y_i changes its value, it will affect the value of all the functions corresponding to the potential advisor and advisee sets $Y_i \cup Y_i^{-1}$.

There is additional information stored in each variable node, as shown in the tables in Figure 6.3. y_i can take different values from Y_i, and the corresponding st_i, ed_i and $g(y_i, st_i, ed_i)$ are pre-computed in stage 1. Here we take l_{ij} as $g(y_i, st_{ij}, ed_{ij})$ when $y_i = j$.

Theoretically, one can incorporate any types of features into the TPFG model. For different kinds of relationships, the constraint can vary according to primary assumptions.

6.1.5 Model inference

To maximize the objective function and compute the ranking score along with each edge in the candidate graph G', we need to infer the marginal maximal joint probability on TPFG, according to Eq. (6.10). We first introduce the algorithm for general factor graph, discuss its deficiency, and then propose our algorithm.

Sum-product + junction tree. There is a general algorithm called *sum-product* [49] to compute marginal function on a factor graph based on message passing. It performs exact inference on a factor graph without cycles. In the sum-product algorithm, the marginal functions of a single variable, a.k.a., messages, are passed between neighboring variable node and function node. Message passing is initiated at the leaves. The process terminates when two messages have passed on every edge. At each variable node, the product of all incoming messages is its marginal function. To compute the marginal maximal probability, we need to change sum-product to max-sum with a logarithmic transformation of the function value. If TPFG is a tree-structured factor graph, the message passing rule will be:

$$\begin{aligned} m_{y_i \to f_j()}(y_i) &= \sum_{j' \in Y_i \cup \{i\}, j' \neq j} m_{f_{j'}() \to y_i}(y_i) && (6.11) \\ m_{f_j() \to y_i}(y_i) &= \max_{\sim\{y_i\}} (\log f_j(y_i, \{y_{i'}\}) + \\ &+ \sum_{i' \in Y_j^{-1} \cup \{j\}, i' \neq i} m_{y_{i'} \to f_j()}(y_{i'})) && (6.12) \end{aligned}$$

where $j' \in Y_i \cup \{i\}, j' \neq j$ represents $f_{j'}()$ is a neighbor node of variable y_i on the factor graph except factor $f_j()$, $\sim \{y_i\}$ represents all variables in $Y = \{y_1, \ldots, y_{n_a}\}$ except y_i.

Unfortunately, TPFG contains cycles. This algorithm cannot be applied directly. One solution to generalize it is a procedure known as *junction tree algorithm* [11] for exact inference. The junction tree is a tree-structured undirected graph generated from arbitrary triangulated dependency graph, and can be solved by sum-product. Nevertheless, the computational cost of the algorithm is determined by the number of variables in the largest clique and will grow exponentially with this number in the case of discrete variables. The process to construct a junction tree alone consumes a lot in both time and space. In practice we found it fails to finish for 6000 variables, not to mention our TPFG has the scale of more than 600,000 variables.

To reduce the computational cost, we can do approximate inference instead of exact inference. A general method *loopy belief propagation* (LBP) [30] simply applies the sum-product algorithm in a cycle-containing graph. It passes message iteratively with flooding schedule. To avoid repetitive information flow for multiple times through the graph, we design a special message passing schedule and the following algorithm according to the special property of TPFG.

New TPFG inference algorithm. The original sum-product or max-sum algorithm meet with difficulty since it requires that each node needs to wait for all-but-one message to arrive. Thus in TPFG some nodes will be waiting forever due to the existence of cycles. To overcome this problem, we arrange the message passing in a mode based on the strict order determined by G'. Each node a_i has a descendant set Y_i^{-1} and an ascendant set Y_i.

The message is passed in a two-phase schema. In the first phase, messages are passed from advisees to possible advisors, and in the second, messages are passed back from advisors to possible advisees. Formally, there are two kinds of messages in the first phase: $m_{f_i() \to y_i}, m_{y_i \to f_j()}$ where $j \in Y_i$. The message from $f_i()$ to y_i is generated and sent only when all the messages from its descendants have arrived. And y_i immediately send it to all its ascendants $f_j(), j \in Y_i$. In phase two, there are also two kinds of messages: $m_{y_i \to f_i()}, m_{f_j() \to y_i}, j \in Y_i$, each of which are along the reverse direction on the edge as in phase 1. The messages are calculated as follows, derived from Eqs. (6.12) and (6.11).

$$
\begin{aligned}
m_{f_i()\to y_i}(x) &= \max_{st_{ki}>ed_{ix},\forall y_k=i}(\log l_{ix} + \\
&+ \sum_{k'\in Y_i^{-1}} m_{y_{k'}\to f_i()}(y_{k'})) \qquad (6.13)\\
m_{y_i\to f_j()}(x) &= m_{f_i()\to y_i}(x) \qquad (6.14)\\
m_{y_i\to f_i()}(x) &= \sum_{j\in Y_i} m_{f_j()\to y_i}(x) \qquad (6.15)\\
m_{f_j()\to y_i}(x) &= \max_{st_{kj}>ed_{jy_j},\forall y_k=j}(\log l_{jy_j}\\
+m_{y_j\to f_j()}(y_j) &+ \sum_{k'\in Y_j^{-1},k'\neq i} m_{y_{k'}\to f_j()}(y_{k'})) \qquad (6.16)
\end{aligned}
$$

After the two phases of message propagation, we can collect the two messages on any edge and obtain the marginal function.

$$
\begin{aligned}
r_{ij} &= \max P(y_1,\ldots,y_{n_a}|y_i=j)\\
&= \exp\left(m_{f_i()\to y_i}(j) + m_{y_i\to f_i()}(j)\right) \qquad (6.17)
\end{aligned}
$$

This algorithm still has redundant storage and computation. The messages sent between function nodes and variables nodes are function values, which need to be stored as vectors. Some messages are never used during the final merge, and some messages are simply transmitted from one variable node to its corresponding function node. We further simplify the message propagation by eliminating the function nodes and the internal messages between a function node and a variable node, and we find it equivalent to a message passing procedure on the homogeneous graph G', *i.e.*, message propagation between authors, and the messages can be stored with each author in two vectors: one sent and one received. The order of messages passed is illustrated by the number on each edge in Figure 6.4. In this way both time and space are saved.

The improved message propagation is still separated into two phases. In the first phase, the messages $\mathbf{sent}_i$ which passed from one to their ascendants are generated in a similar order as before. In the second, messages returned from ascendants $\mathbf{recv}_i$ are stored in each node. After the two phases, each node collects the two vectors to generate the final ranking score. The derived rules are as follows.

$$\begin{aligned}
\mathbf{sent}_{ij} &= \log l_{ij} + \sum_{k \in Y_i^{-1}} \max_{st_{kx} > ed_{ij} \text{ or } x \neq i} \mathbf{sent}_{kx} && (6.18)\\
\mathbf{recv}_{ij} &= \max_{j' \in Y_j, ed_{jj'} < st_{ij}} (\mathbf{recv}_{jj'} + \log l_{jj'} + \\
&+ \sum_{k \in Y_j^{-1}, k \neq i} \max_{x \in Y_k, st_{kx} > ed_{jj'} \text{ or } x \neq j} \mathbf{sent}_{kx}) \\
&+ \sum_{x \in Y_i, x \neq j} \max_{j' \in Y_x} (\mathbf{recv}_{xj'} + \\
&+ \sum_{k \in Y_x^{-1}, k \neq i} \max_{x' \in Y_k, st_{kx'} > ed_{xj'} \text{ or } x' \neq x} \mathbf{sent}_{kx'}) && (6.19)\\
r_{ij} &= \exp(\mathbf{sent}_{ij} + \mathbf{recv}_{ij}) && (6.20)
\end{aligned}$$

In the new algorithm, the message propagation can be done by using a stack-queue. In phase 1, each node will enter the queue once and the vector $\mathbf{sent}_i$ for them is computed one by one. In phase 2, we scan the queue from the tail back to the head, *i.e.*, treat it as a stack, and compute $\mathbf{recv}_i$. Then we can normalize the results and collect them to get the ranking score. By using $\mathcal{O}(|\mathcal{E}_s'|)$ space, the running time of the algorithm can be reduced to $\mathcal{O}(\sum_{i=1}^{n_a} \delta_i \delta_i')$, where δ_i and δ_i' are the in-degree and out-degree of each node a_i on graph G', respectively. As long as G' is sufficiently sparse, the maximal degree of the node can be seen as constant C and the complexity is further reduced to $\mathcal{O}(n_a)$.

6.1.6 Experiments

Datasets. We use the DBLP Computer Science Bibliography Database as the dynamic collaboration data set H to infer the advisor-advisee. It consists of 654,628 authors and 1,076,946 publications with time provided (from 1970 to 2008). To test the accuracy of the discovered advisor-advisee relationships, we adopt three data sets: One is manually labeled by looking into the home page of the advisors, and the other two are crawled from the Mathematics Genealogy project[2] and AI Genealogy project[3]. We refer to them as MAN, MathGP and AIGP respectively. They only paretically cover the authors in DBLP. We further separate MAN into three sub data

[2] `http://www.genealogy.math.ndsu.nodak.edu/`

[3] `http://aigp.eecs.umich.edu/`

sets: Teacher, PhD and Colleague. Teacher contains all kinds of advisor-advisee pairs, while PhD only contains graduated PhDs pairing with their advisors. Colleague contains colleague pairs which are negative samples for advisor-advisee relationship. And we use these data to generate random data sets for test. See Table 6.1 for details.

Method. We compare the proposed TPFG with the following baseline methods:

- Sum-Product+Junction Tree (JuncT). It computes the exact joint probability as the ranking score.
- Loopy Belief Propagation (LBP). It employs an approximate algorithm for inference.
- Independent Maxima (IndMAX). It computes the maximal local likelihood for each variable independently.
- SVM. It is a supervised approach and requires labeled pairs, both positive and negative, as training data.
- RULE. For each author, from all the collaborators that satisfy Assumption 6.2, choose the one with most coauthored papers.

Effect of network structure

Using DFS with a bounded maximal depth d from the given set of nodes, denoted as DFS-d, we can obtain closures with controlled depth for a given set of authors to test. When d increases, the subnetwork grows larger until it is already the complete closure, *i.e.*, the maximal connected subgraph of H containing the given set. We run TPFG on these closures and plot the ROC curves.

From Figure 6.5(a) we see that for closures with different depths, TPFG achieves better accuracy when the depth increases, and they all outperform the IndMAX method by more than 5% in AOC. And on the complete closure TPFG reaches the same accuracy as on the whole network since disconnected components will not affect each other.

On these various scaled subnetworks, TPFG achieves different level of approximations to the optimal global joint probability on the whole network. To compare it with the exact maximal joint probability and other approximate algorithm, we show the result on a small graph due to limitations of JuncT and LBP (see Section 6.1.5). The small graph is constructed by extracting the nodes in PhD and their advisors, and then building 1-closure of it. It consists of 1310 nodes. From Figure 6.5(b) we find that in the small graph TPFG approximates well to the exact inference algorithm JuncT(AOC difference < 0.01), and outperforms LBP by 16.9%.

Effect of training data

Support Vector Machines(SVMs) are accurate supervised learning approaches and shown to be successful in syntax-based relation mining[21]. If we treat advisor-advisee pairs as positive examples and non advisor-advisee pairs as negative examples, we can reduce advisor mining to a classification problem on the ordered pairs (a_i, a_j). In this setting it requires to define some features for each pair of coauthors, and train the classifier by feeding both positive and negative samples. For fair comparison with our probabilistic model, we combined Kulczynski and IR measures with what were used in [97] as features.

Direct application of SVM only shows whether a given pair is an advisor-advisee pair, and it is often the case an author is predicted to have multiple advisors, 1001 out of 2657 for TEST1, for example. Thus we examine the probabilistic scores in the test data, and rank them to draw the ROC curve. TPFG is 4.2% and 2.4% higher in AOC than SVM in TEST1 and TEST2 respectively.

Although in this work we define our model as an unsupervised learning approach, it can utilize the training data in a simple way. We can optimize the parameter θ in the $P@k, \theta$ as we mentioned in Section 6.1.1 according to certain criteria such as achieving best information gain on the training data. Then we use the trained parameters to do predictions on test data. Table 6.1 shows the improvement by utilizing the training data. After this simple training, TPFG can reach an accuracy of 84% to 91%.

SVM actually makes a supervised combination of all the assumptions and rules used in TPFG. The difference lies in that it does not explore the constraint and dependency replying on the whole network structure. It does a fairly good job, but still 5-10% worse than optimized TPFG. In conclusion, TPFG can achieve comparable or even better accuracy compared with a supervised method. When parameters are adjusted with training data, its accuracy can be further improved by around 3%.

Case study

With case study, we find that TPFG can discover some interesting relations beyond the "ground truth" from single source. Table 6.2 shows some examples. Our ranking results provided with advising time facilitate finding such kind of advising relations, which cannot be easily discovered by referring to Genealogy projects. The mean of deviation of estimated graduation time to the labeled time on the test data sets is $1.76 \sim 1.78$.

We find that at least 40% of the error is contributed by name ambiguity. For example, if we try to find the advisor for "Joseph Hellerstein", our algorithm returns wrong results. If we distinguish "Joseph M. Hellerstein" and his publications properly, our algorithm is able to find the 'half' right answer Michael Stonebraker ranked top 1. The answer is half right because there is a co-advisor Jeffrey F. Naughton, who is also ranked high in top 15%. Duplication are even more common among Chinese names. Other reasons for false negatives include that one researcher collaborated with multiple advisors or that one coauthored fewer papers with the advisor. The latter case happens more often for older researchers, for whom the publication data are not as complete as nowadays. In those situations, it is almost impossible to find credible advisor-advisee relationships merely based on their publication records. The inference algorithm can find typical cases but will miss such atypical cases.

Scalability performance

TPFG is shown to be scalable in Figure 6.6. With regard to the inference time, TPFG costs only 13 seconds on the whole DBLP dataset. As a classification approach, SVM's scalability is related to the size of training data. As an example, the feature computation takes one hour and a half, and the model inference takes 31 seconds for Train1 and 6min 26s for Train2.

6.2 Supervised setting

In the supervised setting, we have more complex features, constraints and knowledge propagation rules with unknown importance. We need to learn their importance from the data. Also, we need to find a generic model to handle different types of signals.

Example 6.1 *Given a set of named people and with knowledge of their ages, nationality and mentions from text, we want to find the* family *relation among them. Besides text features such as the coreferential link between two names with family keywords, we have constraints like:* (1) for two persons to be siblings, they should have the same parent; and (2) if A is B's parent, then A is unlikely to be C's parent if B and C have different nationalities. *The output should be a family tree or forest of these people.*

Example 6.2 *In an online forum, we want to predict the replying relationship among the posts within the same thread, with knowledge of the content, author and the posting time. Every post replies to one earlier post in the same thread, except for the first one. One intuition is that* a post will have similar content with the one it replies to*; yet another possibility is* two similar posts may reply to a common post*. The output is a tree structure of each threaded discussion.*

We used these examples as cases for study, and developped a generic discriminative model CRF-Hier.

6.2.1 Conditional random field for hierarchical relationship

We model the joint probability of every possible relationship $(v_i, v_j) \in \mathcal{E}$ being a truly existent relationship in R. We use an indicator variable x_{ij} for the event "$(v_i, v_j) \in R$", *i.e.*, $x_{ij} = 1$ if $(v_i, v_j) \in R$, and 0 if not. As we have analyzed, the inference of the relationship for some pairs are not independent. Suppose we have evidence that two people v_2 and v_4 are not siblings, we may not expect the two events "$(v_2, v_1) \in R$" and "$(v_4, v_1) \in R$" to happen together. We formalize that intuition as a Markov assumption that events involving common objects are dependent.

Assumption 6.3 *Two events "$(v_a, v_b) \in R$" and "$(v_c, v_d) \in R$" correspond to connected random variables in a Markov network if and only if they share a common node, i.e., one of the following is true: $v_a = v_c$, $v_b = v_c$, $v_a = v_d$ or $v_b = v_d$. (Figure 6.8)*

It immediately follows that the Markov network can be derived from the candidate graph G by having a node for every edge in G and connecting nodes that represent two adjacent edges in G, namely the *line graph* of G.

The conditional joint probability is formulated as

$$p(X|G,\Theta) \propto \exp\Big(\sum_{k=1}^{K}\theta_k F_k(X,G) + C(X,G)\Big) \tag{6.21}$$

where $\{F_k(X,G)\}_{k=1}^{K}$ is a set of features defined on the given candidate graph G and the indicator variables $X = \{x_{ij}\}$; $\{\theta_k\}_{k=1}^{K}$ are the weights of the corresponding features. C is a special feature function to enforce the hierarchy constraint $\sum_{(v_i,v_j)\in\mathcal{E}} x_{ij} \leq 1$.

$$C(X,G) = \begin{cases} -\infty & \exists i, \sum_{(v_i,v_j)\in\mathcal{E}} x_{ij} > 1 \\ 0 & o.w. \end{cases} \tag{6.22}$$

Any other hard constraints can be encoded in the same manner.

Thus, once we learn the weights $\{\theta_k\}_{k=1}^{K}$ from training data, the relation inference task could be formulated as a maximum a posterior (MAP) inference problem: for each given candidate DAG G, we target for the optimal configuration of the relationship indicator X^*, such that,

$$X^* = \arg\max_{X\in\mathscr{X}} p(X|G,\Theta) \tag{6.23}$$

where $\mathscr{X}$ is the set of all the possible configurations, *i.e.*, the search space. Since every x_{ij} can take 0 or 1, the size of the space is $2^{|\mathcal{E}|}$.

Such a formulation keeps the maximal generality, while it poses great challenge to solve the combinatorial optimization problem. We improve it in two ways.

First, we explore the form of feature functions F_k. Each of them can be defined on all variables, and then the computation of every function value relies on the enumeration of X in $\mathscr{X}$. However, since we have made the Markov assumptions, the dependency can be represented by a factor graph, and each feature function can be decomposed into the potential functions over cliques of the graph (Hammersley-Clifford theorem [36]). Moreover, we can restrict the potential functions to have interactions between at most a pair of random variables x_{ij} and x_{st}. In fact, we have the following claim: Any factor graph over discrete variables can be converted to a factor graph restricting to pairwise interactions, by introducing auxiliary random variables. This property is found by Weiss and Freeman [94]. Although the generic procedure of conversion may introduce additional variables, we will show that quite a broad range of features can be materialized in as simple forms as pairwise potentials without the help of auxiliary variables in the next section. Here we factorize the constraint H to exemplify the philosophy: $C(X, G) = \prod_{e_{is}, e_{it} \in \mathcal{E}} h(x_{is}, x_{it})$, where $h(x_{is}, x_{it}) = -\infty$ if $x_{is} = x_{it} = 1$, and 0 otherwise.

Next, we try to reduce the number of variables and constraints. To leverage the constraint that one node has at only one parent and the assumption that the candidate graph G is a DAG, we introduce a variable y_i to represent the parent of v_i, *i.e.*, $y_i = j$ if and only if $e_{ij} = (v_i, v_j)$ is an instance in a hierarchical relationship R. Given the DAG, the problem is equivalent to the task of predicting y_i's value from Y_i.

Hypothesis 6.3 implies the existence of two kinds of dependencies: two variables y_i and y_j where v_j is a candidate parent of v_i; two variables y_s and y_t such that v_s and v_t share a common candidate parent v_m. With this formulation, the constraint C is not needed any more, and the objective function has the following form:

$$\begin{aligned} p(Y|G, \Theta) \;\; \propto \;\; & \exp\Big(\sum_{k \in I_1} \sum_{y_s \in S'_k} \theta_k g_k(y_s|Y_s) \\ & + \sum_{k \in I_2} \sum_{(y_i, y_j) \in P'_k} \theta_k f_k(y_i, y_j|Y_i, Y_j)\Big) \end{aligned} \tag{6.24}$$

where I_1 and I_2 are the index sets of features that can be decomposed into singleton potential and pairwise potential functions respectively, and S_k and P_k are the decomposed singleton and pairwise cliques for the k-th feature.

As an example, for the candidate graph G in Figure 6.9 we will build a factor graph like the one on its right. Four nodes v_1, v_2, v_3, and v_4 have four parent variables y_1 to y_4, while the range for them is $Y_1 = \{v_1\}, Y_2 = \{v_1, v_2\}, Y_3 = \{v_3\}$, and $Y_4 = \{v_1, v_3\}$. For each variable there can be one or more singleton potential functions. For each pair of directly linked nodes, we have a pairwise potential function f_4 in this example (we omit the one between y_1 and y_4). v_2 and v_4 have a common parent candidate, so there is one pairwise potential f_5 defined on y_2 and y_4.

We name our model CRF-Hier as it is a conditional random field optimally designed for the hierarchical relationship learning problem.

We give several examples for specific potential definitions.

Example 3. The names of a parent and a child should appear in the same sentence (support sentence) together with family relationship words like "son, father".

$g(y_i|Y_i) = \#\text{support sentences of } (v_i, v_{y_i})$.

Example 4. Siblings have common parent.

$f(y_i, y_j|Y_i, Y_j) = [y_i = y_j]p(v_i \text{ and } v_j \text{ are siblings})$. where we use Iverson bracket to denote a number that is 1 if the condition in square brackets is satisfied, and 0 otherwise. In this case, $[y_i = y_j] = 1$ if $y_i = y_j$ and 0 otherwise.

Example 5. Two people of the same age are unlikely siblings.

$f(y_i, y_j|Y_i, Y_j) = [y_i = y_j][v_i\text{'s age} = v_j\text{'s age}]$.

Note that these statements are not always true, *e.g.*, twins can be siblings. But the advantage of our framework is that it does not enforce them to be true. The indicator in Example 5 takes value 0 or 1, and the weight of the corresponding feature controls how much we trust this rule. Eventually the compatibility with this rule plays a factor as the product of potential function and feature weight in the additive log-likelihood.

We see that these different features, constraints and propagation rules can be encoded in a unified form; we just need to define the potential for each of them. We discuss how to systematically design potentials in the next subsection, followed by the inference and learning algorithm.

6.2.2 Potential function design

We have restricted our focus to the features that can be decomposed into either singleton potential or pairwise potential functions. We summarize the potential types with domain-independent cognitive meanings so that one can design domain-specific potentials with the map.

We start from important singleton potentials.

- *homophyly.* The first kind, and probably most widely applicable, is a similarity measure between two objects. The assumption here is that the filiation connects to homophily, *e.g.*, content similarity, interaction intensity (*e.g.*, the frequency of telephone calls in unit time), location adjacency, time proximity (*e.g.*, whether two documents are published within a short period).

This kind of similarity measure *sim* is symmetric, and there are numerous metrics we can use. The potential function has the form $g(y_i|Y_i) = sim(v_i, v_{y_i})$.

- *polarity.* The second kind, almost equally important, is an asymmetric similarity measure. It is used to measure the dominance of certain attributes of the parent on the child, *e.g.*, authority difference, bias of interaction tendency (*e.g.*, whether A writes many more emails to B than B to A), the degree of conceptual generalization/specialization. Such measure *asim* quantifies the partial order in terms of polarity between linked nodes, in the form of $g(y_i|Y_i) = asim(v_i \to v_{y_i})$.
- *support pattern.* The third kind of potentials characterize the preference to certain patterns involving a pair of nodes with filiation. We can define a potential based on the number of pattern occurrences: $g(y_i|Y_i) = |SP(v_i, v_{y_i})|$, where SP denotes the support pattern set.

The pairwise potentials are responsible for the knowledge propagation as well as the restrictive dependencies.

- *attribute augmentation.* One intuition for the knowledge propagation is that one node can inherit attributes from its parent or child to augment its own. In our model this can be realized by defining a pairwise potential $f(y_i, y_j|Y_i, Y_j) = [y_i = j]sim(v_i, v_{y_j})$. It can be elaborated in two ways: knowing that the parent of v_i is v_j, v_j's parent will tend to choose a similar one with v_i; or given that the parent of v_j is v_{y_j}, it affects the decision of its child towards inheriting attributes from its parent. By replacing the boolean indicator $[y_i = j]$ with a weighting function of v_i and v_j we can control the extent to which we propagate the attribute.
- *label propagation.* Sometimes we can measure how likely two nodes share the same parent, so the label of one's parent can be propagated to similar nodes with a function like $f(y_i, y_j|Y_i, Y_j) = [y_i = y_j]sim(v_i, v_j)$. Given v_i's parent v_{y_i}, the more similar v_j is with v_i, the larger contribution to the joint likelihood this function will have via setting v_j's parent to be the same, *i.e.*, $y_j = y_i$.
- *reciprocity.* This kind of potentials can handle a more complicated pattern that occurs in child-parent and parent-child pairs alternatively. For example, "if author A often replies author B, then author B is more likely to reply author A". For such kind of rule, we seem to need a big factor function like $F(y_i|G, Y \setminus y_i) = \sum_{A,B}([a_i = B][a_{y_i} = A]\sum_{a_j=A}[a_{y_j} = B])$, where a_i stands for v_i's author. It requires all labels to be known. Fortunately, we find that rules in this form

have equivalent decomposed representation. The above rule can be decomposed into pairwise potentials $f(y_i, y_j|Y_i, Y_j) = [a_i = a_{y_j}][a_{y_i} = a_j]$.

For a specific application we can encode an arbitrary number of potentials of each type. In experiment part we will use three real-world examples to demonstrate that. In principle, one can apply frequent pattern mining or statistical methods to find distinguishing features in each category. When features in multiple categories are used, the model needs to be learned to reach a compromise.

6.2.3 Model inference and learning

Given a training set of network $G = \{\mathcal{V}, \mathcal{E}\}$ with labeled instances with hierarchical relationship L, we need to find the optimal model setting $\Theta = \{\theta_k\}_{k=1}^{K}$, which maximizes the conditional likelihood defined in (6.24) over the training set. Let $Y^{(o)}$ and T be the variable sets and assignments corresponding to labeled relation instances L, and $Y^{(u)}$ be the unknown labels in training data. Their union is the full variable set Y in training data. The log-likelihood of the labeled variables could be written as:

$$\begin{aligned} L_\Theta =& \log p(Y^{(o)} = T|G, \Theta) \\ =& \log \sum_{Y^{(u)}} \exp\left(\Theta^t \mathbf{F}(Y|_{Y^{(o)}=L}, G)\right) - \log Z(G, \Theta) \end{aligned} \tag{6.25}$$

where $\mathbf{F}(Y, G)$ is the vector representation of the feature functions, $Z(G, \Theta) = \sum_Y \exp(\Theta^t \mathbf{F}(Y, G))$.

To avoid overfitting, we penalize the likelihood by L2 regularization. Taking the derivative of this object function, we could get:

$$\begin{aligned} \nabla L_\Theta &= E_{p(Y^{(u)}|G,\Theta,Y^{(o)}=T)} \mathbf{F}(Y|_{Y^{(o)}=T}, G) \\ &- E_{p(Y|G,\Theta)} \mathbf{F}(Y, G) - \lambda\Theta \end{aligned} \tag{6.26}$$

When the training data are fully labeled, $Y^{(u)} = \emptyset$, and Eq. (6.26) becomes simply:

$$\nabla L_\Theta = \mathbf{F}(Y = T, G) - E_{p(Y|G,\Theta)} \mathbf{F}(Y, G) - \lambda\Theta$$

The first part is the empirical feature value in the training set, the second part is the expectation of the feature value in the given training data, and the last part is the L2-regularization. Given that the expectation of the feature value can be computed efficiently, L-BFGS [16] algorithm can be employed to optimize the objective function Eq. (6.25) by the gradient, although it is not convex with respect to Θ, and we can expect to find a local maximum of it. When there are multiple training networks, we train the model by optimizing the sum of their log likelihood.

When the feature weights are fixed, the learning process requires an efficient inference algorithm for marginal probability of every clique: singletons and edges where we define our potentials. The prediction, however, requires MAP inference to find maximal joint probability. Loopy belief propagation (LBP) [30] *etc.*, have been shown effective to achieve empirically good inference and also more scalable than general-purpose LP solvers (CPLEX).

In the toy model in Figure 6.9, there are 5 different potentials g_1, g_2, g_3, f_4, f_5. The algorithm will learn the weight vector $\Theta = \{\theta_1, \ldots, \theta_5\}$ from training data, and then find the optimal value of $Y = \{y_1, \ldots, y_4\}$ to predict the structure.

6.2.4 Experiments

Evaluation measure. Few studies have been carried out on how we should evaluate the quality of hierarchical relationship prediction. Accuracy on the predicted parent y_i (A_{par}), and the accuracy on the predicted relation pairs x_{ij} (A_{pair}), are two most natural evaluation criteria; they or their variants (Precision, Recall, *etc.*) are employed by most previous studies [73, 99, 83]. However, such measures only evaluate the prediction variables on each node or each edge in an isolated view, missing some aspects of the comprehensive goodness of structure. We take an example to illustrate. Figure 6.10 lists a ground-truth structure and several different reconstruction results. Both (b) and (c) have the same A_{par} and A_{pair} because only one node has incorrect predicted parent. However, the chain is quite different from the gold standard tree with two branches, and result (c) should be regarded closer to the ground truth. So we can see that one mistake may not only affect the parent of one node, but also deviate the shaping statistics of other nodes (*e.g.*, in their degrees, number of ancestors and descendants). In other words, different edges have different importance in preserving the shape of the tree, which is not reflected by the unweighted judgment of each prediction. Tree similarity measures for ontologies such as tree edit distance [10] are neither desirable because the edit operations do not apply here, and the computation of them has biquadratic complexity w.r.t. the tree size.

We define a set of novel measures for quantitative evaluation of the quality of hierarchical relationship prediction, which can be computed in linear time. Formally, let T be the ground-truth structure for n linked objects $V = \{v_1, \ldots, v_n\}$, and Y the prediction. We evaluate how well the structure is preserved in prediction by examining two additional aspects: the ancestors and the path to root.

- Precision and recall of ancestors P_{anc}, R_{anc}.

$$P_{anc} = \frac{\sum_{i=1}^{n} \delta[Anc_i(Y) \subset Anc_i(T)]}{n} \tag{6.27}$$

$$R_{anc} = \frac{\sum_{i=1}^{n} \delta[Anc_i(T) \subset Anc_i(Y)]}{n} \tag{6.28}$$

  where $Anc_i(Y)$ and $Anc_i(T)$ stand for the ancestor set of node v_i in prediction and in ground truth respectively.

- Accuracy of the path to root, A_{path}.

$$A_{path} = \frac{\sum_{i=1}^{n} \delta[path_i(Y) = path_i(T)]}{n} \tag{6.29}$$

  where $path_i(Y)$ and $path_i(T)$ is the path from node v_i to its root in prediction and in ground truth respectively. A_{path} measures whether we can trace from each particular node to root without any mistake, thus it is the most strict measure.

As a commonly used compromise between precision and recall, we can also define F-value for the proposed measure for ancestors as the harmonic mean of precision and recall. For the example in Figure 6.10, the proposed metrics have the values as shown in Table 6.4. We can see that although (b) and (c) have the same accuracy on the parent prediction, three of the four proposed measures all imply the inferred structure in (c) has better quality than the chain. Also, we notice that A_{path} is a most strict measure, and in that measure (b) is even worse than (a). That implies one predicted structure may be good in some aspect but bad in others. This reaffirms the necessity of using multiple measures for the tree structure evaluation.

The last thing we want to point out is that the hardness of the inference problem highly depends on the number of candidate parents of each node. The random guess will have much lower performance than 0.5 in most cases, even when each node only has two or three candidate parents.

Algorithm setting. We observe no obvious difference for the several variants of belief propagation algorithms. The learning results are based on LBP, and the L2 penalty parameter is $\lambda = 2$ when there is no further specification. All the features are normalized into $[-1, 1]$.

Uncovering family tree structure

We apply our method to an entity relation discovery problem. In the Natural Language Processing (NLP) community, it is sometimes studied as a slot filling task [44], *i.e.*, to answer questions like "who are the top employees of IBM". Some relation types satisfy our definition of hierarchical relationship, *e.g.*, manager-subordinate, parent_organization-subsidiary, country-state-city. We take the family relation (parent-child) as the case to study, and try to answer the following two questions: 1) whether the proposed method works better than the state-of-the-art NLP approaches to generic entity relation mining; and 2) how good a joint model is compared to a model that does not handle dependency rules, or uses the rules just for post processing.

For clear demonstration, we define the task as automatically assembling the family tree from a set of named person entities. These named entities were extracted from two famous American families, *Kennedy family* and *Roosevelt family*, as listed in Wikipedia and Freebase [15]. We design potentials according to the map in Section 6.2.2. Table 6.5 lists the potentials we used. Given any pair of named entities we first collected all the Web context sentences (snippets) returned by Yahoo! search engine. Then we encoded features based on analysis of these snippets such as co-occurrence statistics. We also encoded additional features of various discovered attributes including residence, age, birth-date, death-date and other family members. The results of random guess reflect the hardness of the problem.

We compare our method with three baselines: (1) NLP, the general relation mining approaches based on NLP techniques described in [19]. More specifically, we applied a pattern matching algorithm to discovering parent-children relation links among entities by analyzing the Web snippets.

(2) Ranking SVM, a robust machine learning technique developed from support vector machine (SVM) for ranking problem, but only singleton features can be handled. (3) Ranking SVM + post processing (PP). For Ranking SVM, we treat each node as a query, and its parent as relevant "document" and all the other candidates as non-relevant "documents". For post processing, we encode some pairwise potentials as global constraints (e.g., one person cannot have multiple parents; siblings should share the same parents) in Integer Linear Programming (ILP), in order to maximize the summation of confidence values from Ranking SVM subject to these constraints. The detailed implementation is described in [51]. Table 6.6 shows the results on two families with 60 and 40 members respectively. We can see that the optimized model for hierarchical relationship discovery performs 2 to 3-fold better in most measures than the general purpose relation miner. Compared with the two-stage method that uses post processing rules, the joint model can better integrate them into the learning and inference framework. The margin is the largest in the most strict measure path accuracy (333%, 133%). That implies our method makes fewer mistakes at the key positions of the tree structure, where the chance of absorbing knowledge and doing regularization is higher. We also find that RankingSVM and RankingSVM + PP beat NLP in A_{par}, but are no better than NLP in $F1_{anc}$ and A_{path} in the first test case. Also, the post processing does not help much because the confidence estimation from the prediction is not always reliable due to noises in prediction features. As one example, the prediction component successfully identified "*William Emlen Roosevelt*" as the parent of "*Philip James Roosevelt*" but with a low confidence; while it mistakenly identified "*Theodore Roosevelt*" as the parent of "*George Emlen Roosevelt*" but with a much higher confidence due to their high mutual information value in Web snippets. Therefore, in the post-processing stage, based on the fact that "*Philip James Roosevelt*" and "*George Emlen Roosevelt*" are siblings, the label propagation rule mistakenly changed the parent of "*Philip James Roosevelt*" to "*Theodore Roosevelt*". In our model, the weights of the local features and the propagation rules are learned from the data, and the global optimization prevents this mistake.

Uncovering online discussion structure

Online conversation structure is a popular topic studied by researchers in information retrieval, since the structure benefits many tasks including keyword-based retrieval, expert finding and question-answer discovery. We perform the study on the problem of finding reply relationship among posts in online forums. The data are crawled from Apple Discussion forum (`http://discussions.apple.com/`) and Google Earth Community (`http://bbs.keyhole.com/`). From each forum we crawled around 8000 threads and each thread contains 6-7 posts on average, although some threads can be as long as containing 250 posts. The posts in each thread can be organized as a tree based on their reply relationship. The task is to reconstruct the reply structure for the threads with no labels, given a few threaded posts with labeled reply relationship. Although we use the data from the forums where the reply relationship is actually recorded by the system, the real application scenario will be predicting the online conversations whose structure is unknown. Therefore, we want to study the following questions: 1) How many labeled data are needed to achieve good performance; and 2) How is the adaptability of the model when trained on one forum while testing on another.

We list the features of each type in Table 6.7. More intuition behind the features can be found in [92]. The competitor we choose is Ranking SVM, used in [73] for this task. Again, it can only handle singleton features. We also compare with a naive baseline that always predicts chain structure.

To answer the first question, we fix the test data of 2000 threads and vary the training data size in two different ways. First, we use all the labels from each thread, but vary the size of training threads from 50 to 2500. Second, we fix the number of training threads as 1000, and change the number of labels we use for each thread from 3 to 11. From Figure 6.11(a), we find that with a small training set, 50 labeled threads, CRF-Hier already achieves encouraging performance. The margin is significant because even the naive baseline of predicting every post to reply to the last post gives 0.74 in $F1_{anc}$ — compared with Ranking SVM's 0.80 and CRF-Hier's 0.86, CRF-Hier doubles the margin of what can be achieved by Ranking SVM from the naive baseline. As more training data is added, the testing performance is relatively more stable than Ranking SVM. Figure 6.11(b) shows that when the labels for each tree is very incomplete, CRF-Hier degrades its performance to its competitor because the pairwise features cannot be well exploited. When the labels become reasonably sufficient (5 posts in this case) to characterize the structural dependency among them, CRF-Hier presents superiority (increase the margin from baseline by 32% in A_{path}). Although

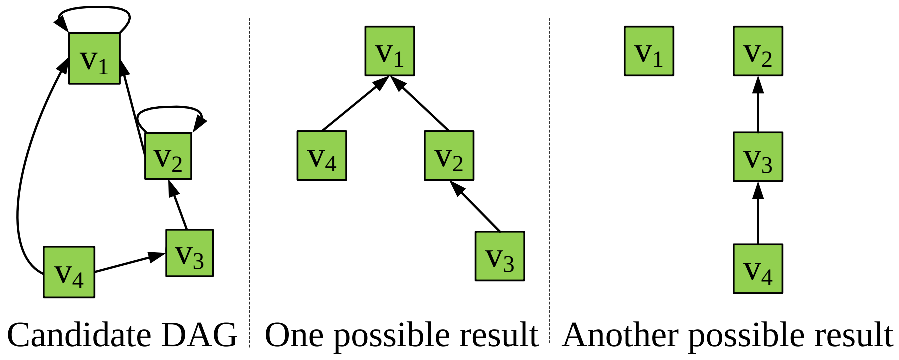


Figure 6.1: An illustration of the problem definition

Table 6.1: Accuracy of prediction by P@(2,θ): $\frac{T}{T+F}$

data set	RULE	SVM	IndMAX		TPFG	
TEST1	69.9%	73.4%	75.2%	78.9%	80.2%	84.4%
TEST2	69.8%	74.6%	74.6%	79.0%	81.5%	84.3%
TEST3	80.6%	86.7%	83.1%	90.9%	88.8%	91.3%

TRAIN1=Colleague(491)+PHD(100)
TEST1=Teacher(257)+MathGP(1909)+Colleague(2166)
TRAIN2=TRAIN3=Teacher(257)+Colleague(2166)
TEST2=PHD(100)+MathGP(1909)+Colleague(4351)
TEST3=AIGP(666)+Colleague(459)
IndMAX,TPFG: left - θ = 3rd quartile of $\{r_{ij}\}$; right - trained

CRF-Hier has more feature weights to learn, the L2 regularization mitigates overfitting, and the model works well even when the training data size is small.

To answer the second question, we randomly select 2,000 threads for training and 2,000 threads for testing from each of the two datasets, and perform a cross domain experiment with the 4 combinations of train/test sets. Apple Discussion is a computer technical forums, while Google Earth focuses on entertainments. From the comparative results in Table 6.8, we can find that with the help of pairwise features, CRF-Hier generalizes better than Ranking SVM which relies on the singleton features only.

In conclusion, CRF-Hier constantly outperforms the baseline when the size or the domain of the training data vary. It has good adaptability to generalize.

Figures and tables

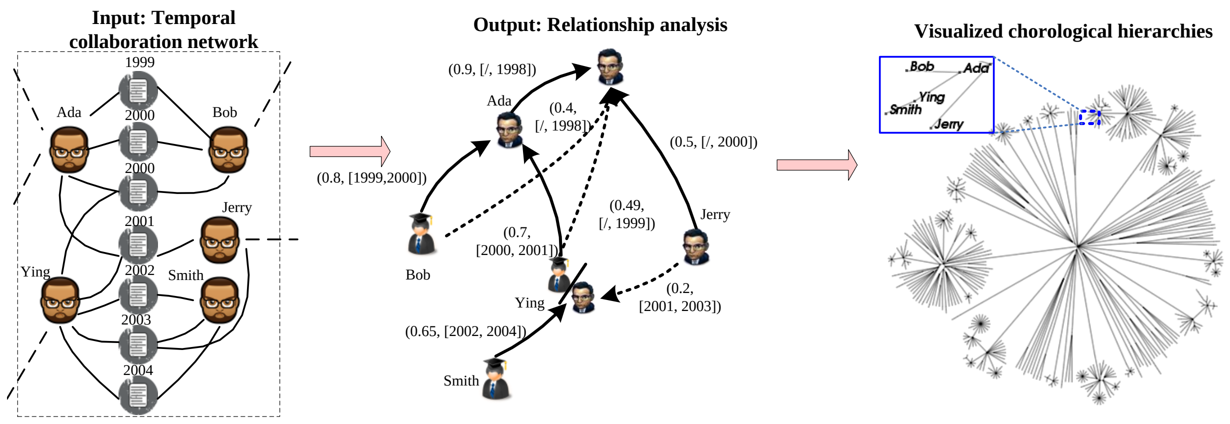


Figure 6.2: Example of advising relationship analysis on the co-author network

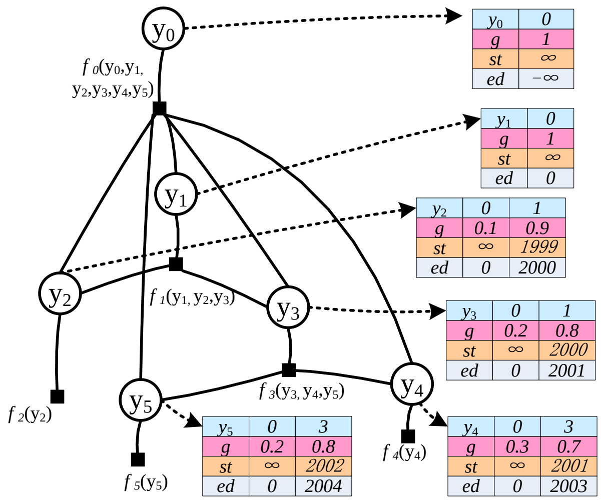

y_0
$f_0(y_0,y_1,y_2,y_3,y_4,y_5)$
y_1
y_2
$f_1(y_1,y_2,y_3)$
y_3
$f_2(y_2)$
y_5
$f_3(y_3,y_4,y_5)$
y_4
$f_5(y_5)$
$f_4(y_4)$
y_0 0; g 1; st ∞; ed −∞
y_1 0; g 1; st ∞; ed 0
y_2 0 1; g 0.1 0.9; st ∞ 1999; ed 0 2000
y_3 0 1; g 0.2 0.8; st ∞ 2000; ed 0 2001
y_5 0 3; g 0.2 0.8; st ∞ 2002; ed 0 2004
y_4 0 3; g 0.3 0.7; st ∞ 2001; ed 0 2003

Figure 6.3: Graphical representation of a time-constrained probabilistic factor graph, where $\{y_0, \ldots, y_5\}$ are hidden variables defined on all nodes; $f_i(.)$ represents a factor function defined on a hidden variable and its potential advisee sets as neighbors

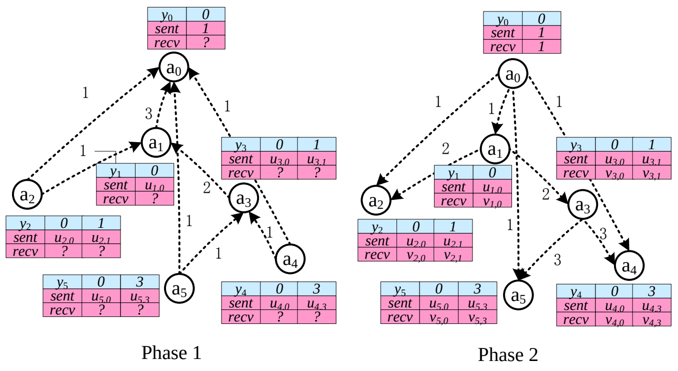
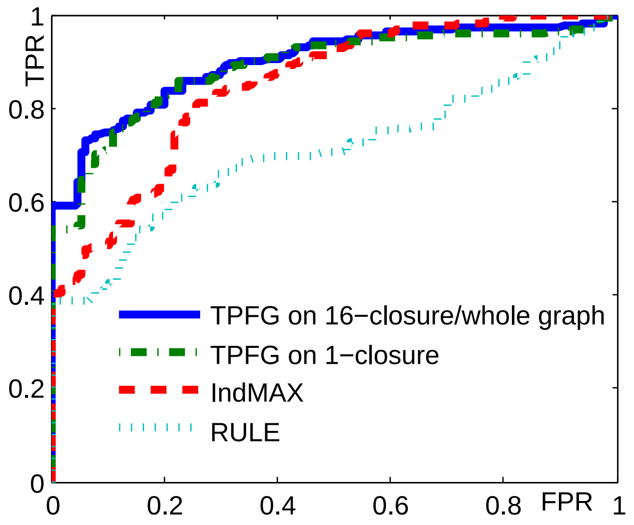
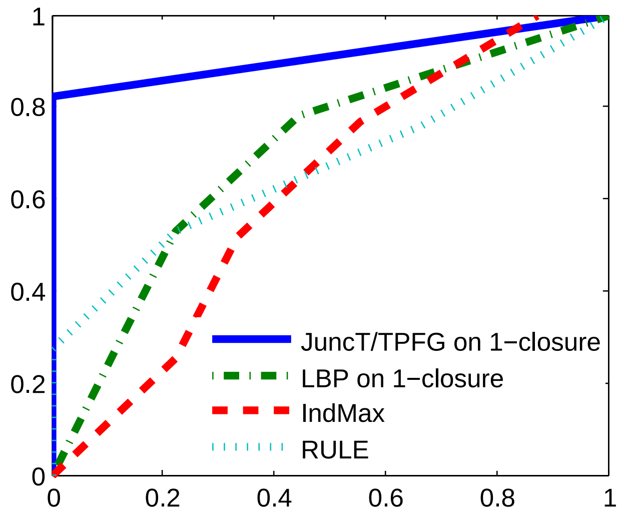

Phase 1
Phase 2
a_0
a_1
a_2
a_3
a_4
a_5
y_0 0; sent 1; recv ?
y_1 0; sent $u_{1,0}$; recv ?
y_2 0 1; sent $u_{2,0}$ $u_{2,1}$; recv ? ?
y_3 0 1; sent $u_{3,0}$ $u_{3,1}$; recv ? ?
y_4 0 3; sent $u_{4,0}$ $u_{4,3}$; recv ? ?
y_5 0 3; sent $u_{5,0}$ $u_{5,3}$; recv ? ?
y_0 0; sent 1; recv 1
y_1 0; sent $u_{1,0}$; recv $v_{1,0}$
y_2 0 1; sent $u_{2,0}$ $u_{2,1}$; recv $v_{2,0}$ $v_{2,1}$
y_3 0 1; sent $u_{3,0}$ $u_{3,1}$; recv $v_{3,0}$ $v_{3,1}$
y_4 0 3; sent $u_{4,0}$ $u_{4,3}$; recv $v_{4,0}$ $v_{4,3}$
y_5 0 3; sent $u_{5,0}$ $u_{5,3}$; recv $v_{5,0}$ $v_{5,3}$

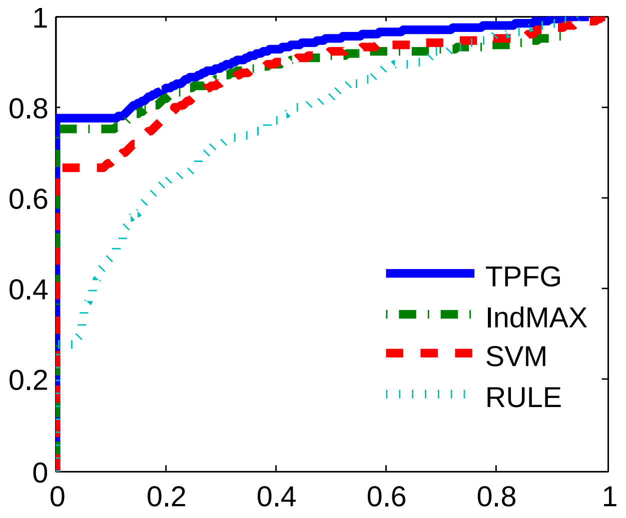
Figure 6.4: The 2-phase message passing schema



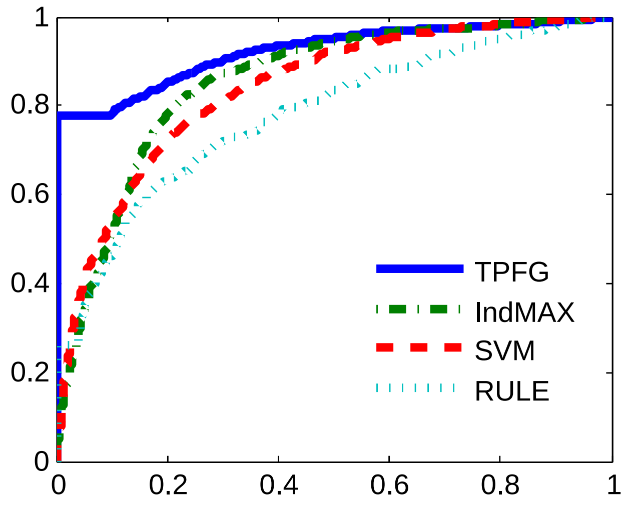
(a) TPFG on different closures of a subnetwork



(b) JuncT, LBP and TPFG on smaller graphs



(c) IndMAX, TPFG and SVM on TEST1



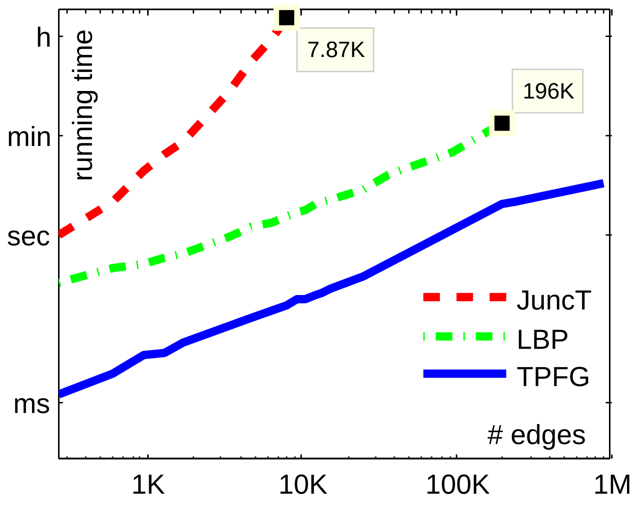
(d) IndMAX, TPFG and SVM on TEST2

Figure 6.5: ROC curves for advisor-advisee prediction

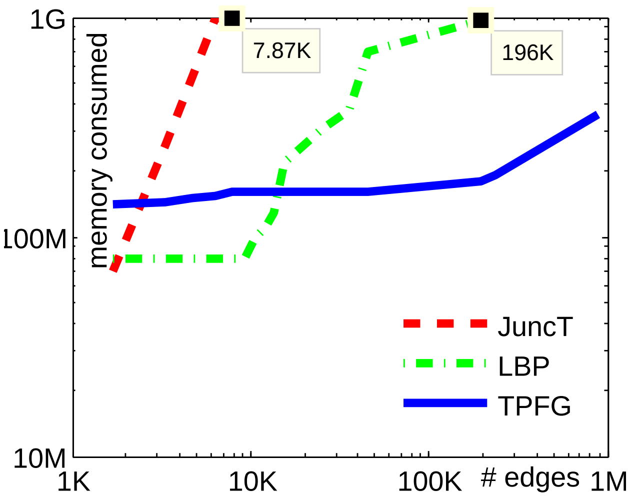
Table 6.2: Examples of mined relations. Time - the estimated advising time; Note - the factual relation and graduation year

Advisee	Top Ranked Advisor	Time	Note
David M. Blei	1. Michael I. Jordan	2001-2003	PhD advisor, 2004
	2. John D. Lafferty	2005-2006	Postdoc, 2006
Hong Cheng	1. Qiang Yang	2002-2003	MS advisor, 2003
	2. Jiawei Han	2004-2008	PhD advsior, 2008
Sergey Brin	1. Rajeev Motwani	1997-1998	‘Unofficial advisor’[4]

[4] cited from a blog of Sergey Brin, who left Stanford to found Google around 1998.

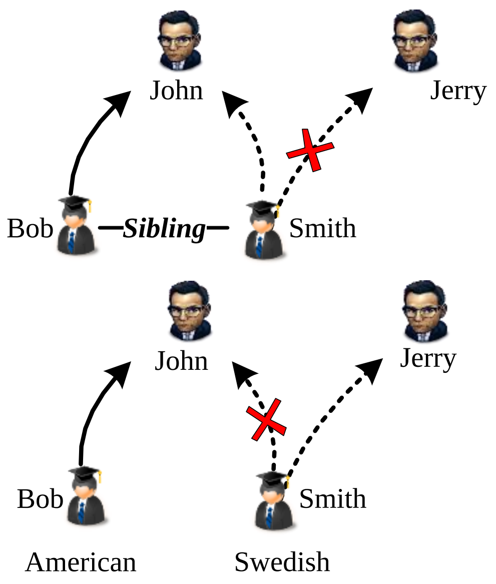


(a) Time

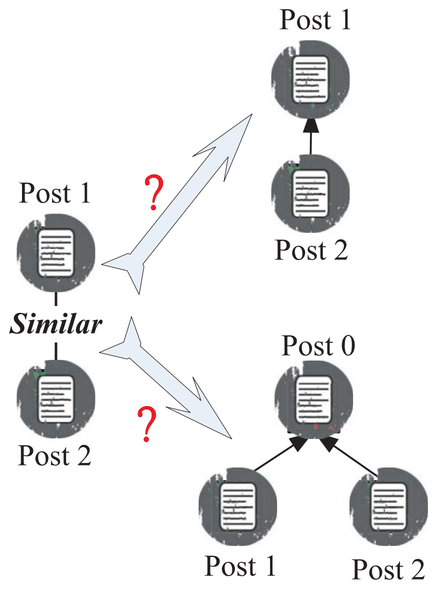


(b) Space

Figure 6.6: Scalability results in log-log scale. JuncT and LBP fail at 10K and 200K respectively due to memory limitation

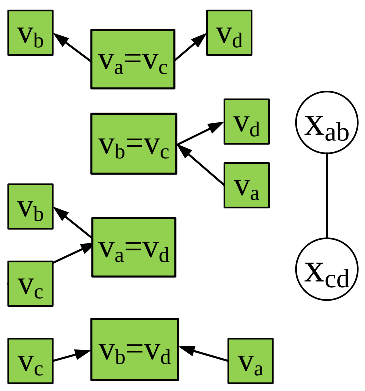


(a) Two soft dependency rules on family tree: the relative importance need be learnt

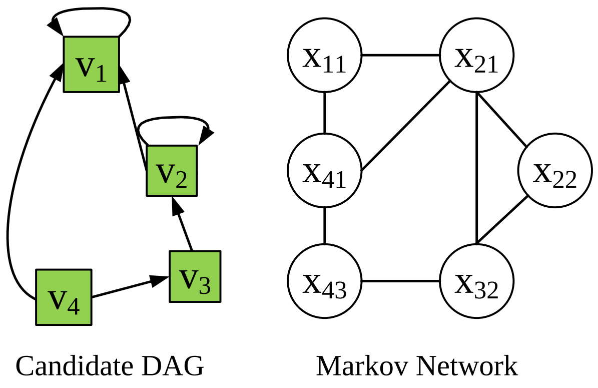


(b) Conflict rules on forum reply structure: similar posts may be filiation or siblings

Figure 6.7: Examples for the dependency among the relations

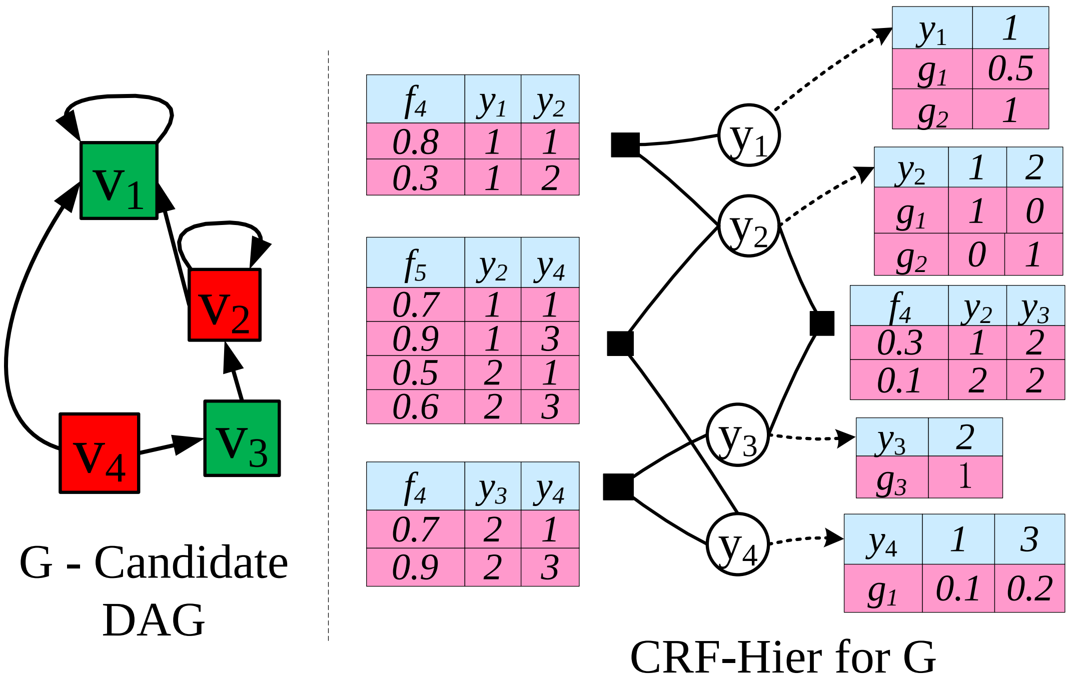


(a) 4 cases for v_aRv_b and v_cRv_d to be directly dependent



(b) The Markov network from the example candidate DAG

Figure 6.8: Illustration of the Markov assumption



f_4	y_1	y_2
0.8	1	1
0.3	1	2

f_5	y_2	y_4
0.7	1	1
0.9	1	3
0.5	2	1
0.6	2	3

f_4	y_3	y_4
0.7	2	1
0.9	2	3

y_1	1
g_1	0.5
g_2	1

y_2	1	2
g_1	1	0
g_2	0	1

f_4	y_2	y_3
0.3	1	2
0.1	2	2

y_3	2
g_3	1

y_4	1	3
g_1	0.1	0.2

Figure 6.9: The graphical representation of the proposed CRF-Hier model for the example in Figure 6.1. All the singleton potentials defined on each variable are listed in the table connected to the related variables by dotted line; pairwise potentials are represented by solid rectangles and tables listing the pairwise function values when the pair of variables take different configurations

Table 6.3: Potential categorization and illustration

Type	Cognitive Description	Potential Definition	Example
homophyly	Parent and child are similar	$g(y_i) = sim(v_i, v_{y_i})$	textual similarity, interaction intensity, spatial and temporal locality
polarity	Parent is superior to child	$g(y_i) = asim(v_i \rightarrow v_{y_i})$	authority, interaction tendency, conceptual extension and inclusion
support pattern	Patterns frequently occurring with child-parent pairs	$g(y_i) = \|SP(v_i, v_{y_i})\|$	contextual pattern, interaction pattern, preference to certain attributes
forbidden pattern	Patterns rarely occurring with child-parent pairs	$g(y_i) = -\|FP(v_i, v_{y_i})\|$	forbidden attribute to share, forbidden distinction
attribute augment	Use inherited attributes from parents or children	$f(y_i, y_j) = [y_i = j] sim(v_i, v_{y_j})$	content propagation for documents, authority propagation for entities
label propagate	Similar nodes share similar parents (children)	$f(y_i, y_j) = sim(v_{y_i}, v_{y_j}) sim(v_i, v_j)$	siblings share common parents, colleagues share similar supervisors
reciprocity	Patterns altering in child-parent and parent-child pairs	$f(y_i, y_j) = sim(v_i, v_{y_j}) sim(v_j, v_{y_i})$	author reciprocity in online conversation - forth-and-back;
constraints	Restrict certain patterns	$f(y_i, y_j) = -\|CP(v_i, v_j, v_{y_i}, v_{y_j})\|$	consistency of transitive property

For conciseness we omit the conditional variable in g and f.
SP = support pattern set, FP = forbidden pattern set, CP = constraint pattern set.

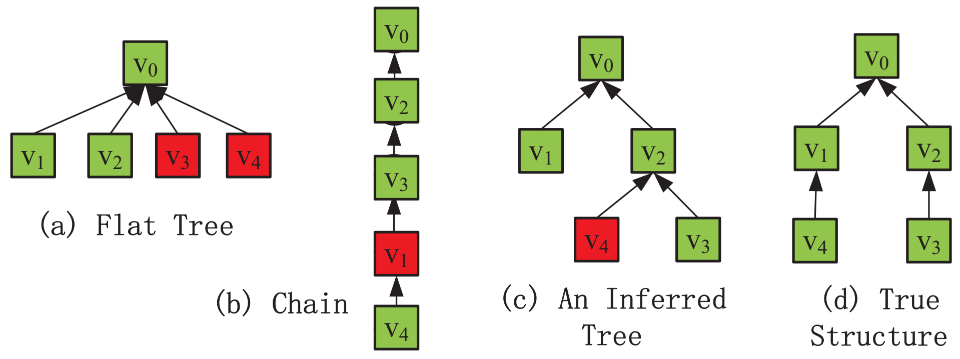


Figure 6.10: Comparison of inferred structures and the gold standard structure. (a)(b)(c) are three possible prediction results for the true structure in (d). Green(light) nodes have right prediction for their parents, while red(dark) nodes have wrong predictions

Table 6.4: Measurement for structures in Figure 6.10

Structure	P_{anc}	R_{anc}	$F1_{anc}$	A_{path}	A_{par}
(a) flat tree	1.00	0.60	0.75	0.60	0.60
(b) chain	0.60	1.00	0.75	0.40	0.80
(c) inferred	0.80	0.80	0.80	0.80	0.80

Table 6.5: Potentials used in family tree construction task

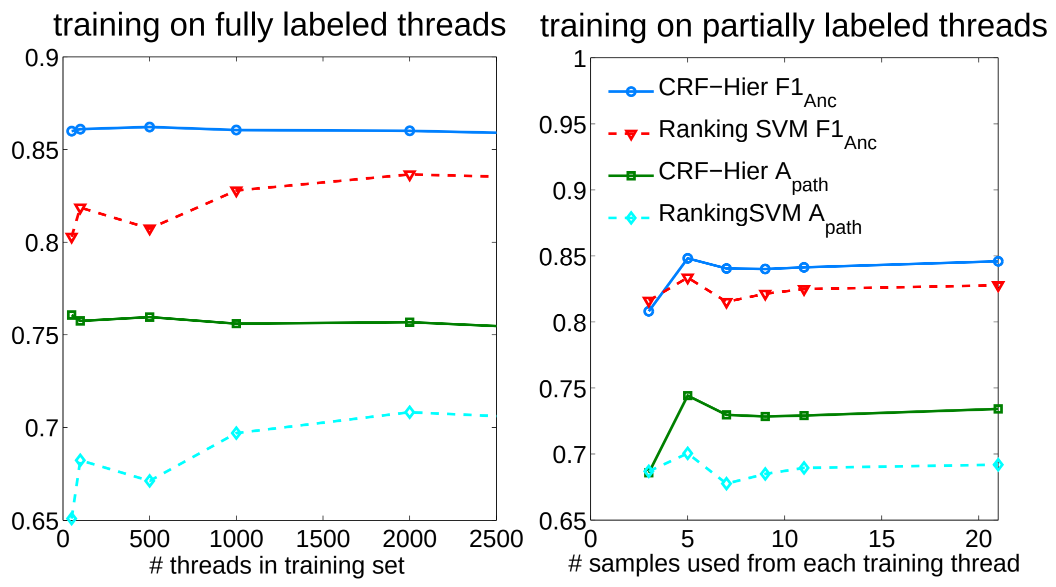
Type	Potential Description
homophily	v_i and v_{y_i} live in the same location? mutual information of two names and family keywords from web snippets
polarity	suffix comparison: Junior, I, III, IV *etc.*
support pattern	co-occurrence of two names and some family keywords from web snippets; parent-child implied by Wikipedia infoboxes
forbidden pattern	child's birth year after parent's death+1; non parent-child implied by Wikipedia
attribute augment	same residence location of one and grandparent
label prop	people who are likely to be siblings share the same parent
constraints	people with the same age get lower possibility to share the same parent

Table 6.6: Prediction performance for family tree

Train/Test	Method	$F1_{anc}$	A_{path}	A_{par}
Train on Kennedy, test on Roosevelt	Random	< 0.01	< 0.01	0.0943
	NLP	0.1146	0.0333	0.1833
	RankingSVM	0.0667	0.0500	0.5000
	RankingSVM+PP	0.0667	0.0500	0.5833
	CRF-Hier	**0.3439**	**0.2167**	**0.7167**
Train on Roosevelt, test on Kennedy	Random	< 0.01	< 0.01	0.1313
	NLP	0.2625	0.1750	0.2500
	RankingSVM	0.4371	0.1500	0.3250
	RankingSVM+PP	0.4750	0.1500	0.3500
	CRF-Hier	**0.4846**	**0.3500**	**0.4000**

Table 6.7: Potentials used in post reply structure task

Type	Features, Rules and Constraints
homophily	tf-idf cosine similarity $\cos(v_i, v_{y_i})$; recency of posting time
polarity	whether v_{y_i} is the first post of the thread; whether v_{y_i} is the last post before v_i
support pattern	whether a_{y_i}'s name appears in post v_i, where a_{y_i} is the author of v_{y_i}
forbidden pattern	an author does not reply to himself
attribute augment	the average content of one post's children is similar to its parent: $[y_i = j] \cos(v_i, v_{y_j})$
label propagate	similar posts reply to the same post: $[y_i = y_j] \cos(v_i, v_j)$
reciprocity	author A replies B, motivated by B replying A: $[a_{y_j} = a_i][a_{y_i} = a_j]$
constraints	one author does not repeat reply to a post; B replying A's post, A does not reply to B's earlier post: $-[t_{y_i} < t_j][a_{y_i} = a_j][a_{y_j} = a_i]$



(a) (b)

Chain structure $F1_{anc} = 0.7435, A_{path} = 0.5647$

Figure 6.11: Performance with varying training data size (Apple)

Table 6.8: Cross domain evaluation (CRF-Hier/Ranking SVM)

	Train \ Test	Apple	Google Earth
$F1_{anc}$	Apple	**0.8476**/0.8136	**0.8233**/0.7855
	Google Earth	**0.8383**/0.8017	**0.8186**/0.8099
A_{path}	Apple	**0.7326**/0.6722	**0.6909**/0.6325
	Google Earth	**0.7143**/0.6548	**0.6797**/0.6610

All improvements are statistically significant with $p < 0.05$

Chapter 7

Scalable and Robust Topic Discovery

All the existing topic modeling approaches, including our hierarchical topic discovery module in Chapter 3, have the following bottlenecks:

1. **Scalability**. The inference methods such as Gibbs sampling are expensive, requiring multiple passes of the data. The number of passes has no theoretical bound, and typically needs to be several hundreds or thousands.
2. **Robustness**. The inference algorithms do not produce a unique solution in nature. The variance of different algorithm runs can be very large especially when the hierarchy is deep. This prevents a user from revising the local structure of a hierarchy (*e.g.*, changing the number of branches of one node).

In this chapter, we take a closer look at this problem, and develop a new solution that overcomes these bottlenecks.[1]

The following summarizes its contributions:

- We propose a new hierarchical topic model, which supports divide-and-conquer inference. We provide theoretical justification of doing so.
- We develop a scalable tensor-based recursive orthogonal decomposition (STROD) method to infer the model. It inherits the nice theoretical properties of the tensor orthogonal decomposition algorithm [6], but has significantly better scalability. To the best of our knowledge, it is the first scalable and robust algorithm for topical hierarchy construction.
- Our experiments demonstrate that our method can scale up to datasets that are orders of magnitude larger than the state-of-the-art, while generating quality topic hierarchy that is comprehensible to users.

[1]Partial content of this chapter is from Wang et al. [91].

7.1 Desired properties

The following are the desired properties for a topical hierarchy construction method.

1. Scalability. The scale of the problem is determined by these variables: the number of documents D, the vocabulary size V, the total length of documents L, the total number of topics T, and the width of the topical hierarchy K. These variables are not independent. For example, the average length of documents L/D should be larger than 1, and the number of documents D is usually much larger than the vocabulary size V. Typically, the number of tokens L is the dominant factor. For scalability the algorithm must have sublinear complexity with respect to L. When the dataset is too large to fit in memory, an ideal algorithm should only perform a small constant number of passes of the data.

2. Robustness. A construction algorithm is robust in the following senses.

Property 7.1 (Robust recovery) *Suppose the data is generated from certain topic word distributions exactly following a generative process, the recovery is robust if the exact distributions can be found when sufficient data are given.*

For example, in Figure 3.1, if sufficient data are generated from the topic word distributions as shown on the right hand side, an robust recovery algorithm should return these distributions rather than other distributions.

In certain scenarios, part of the constructed hierarchy needs to be revised to customize for users' need. For example, one may want to change the number of branches or height of a subtree.

Property 7.2 (Robust revision) *The revision to a subtree $\mathcal{T}(t)$ rooted at topic t is robust, if every topic t' not in the subtree $\mathcal{T}(t)$ remains intact word distribution in the returned hierarchy.*

In Figure 3.1, if one wants to partition topic $t1$ into 3 subtopics instead of 2, but also wants to keep other parts of the tree intact, a robust algorithm should not change the output of topic $t2, t3$ and $t6$ to $t10$. This property assures that the local change to a large hierarchy doesn not alter the remainder of the tree.

7.2 Latent Dirichlet allocation with topic tree

In our model, every document is modeled as a series of multinomial distributions: one multinomial distribution for every non-leaf topic over its child topics, representing the content bias towards the subtopics. For example, in Figure 3.1, there are 4 non-leaf topics: $o, o/1, o/2$ and $o/3$. So a document d is associated with 4 multinomial topic distributions: $\theta_{d,o}$ over its 3 children, and $\theta_{d,o/1}, \theta_{d,o/2}, \theta_{d,o/3}$ over their 2 children each. When the height of the hierarchy $h = 1$, it reduces to the flat LDA model, because only the root is a non-leaf node. Each multinomial distribution $\theta_{d,t}$ is generated from a Dirichlet prior α_t. $\alpha_{t,z}$ represents the corpus bias towards z-th child of topic t, and $\alpha_{t,0} = \sum_{z=1}^{C_t} \alpha_{t,z}$.

For every leaf topic node t, $C_t = 0$, and a multinomial distribution ϕ_t over words is generated from another Dirchlet prior β. These word distributions are shared by the entire corpus.

To generate a word $w_{d,j}$, we first sample a path from the root to a leaf node $o/z_{d,j}^1/z_{d,j}^2/\cdots/z_{d,j}^h$. The nodes along the path are sampled one by one, starting from the root. Each time one child $z_{d,j}^k$ is selected from all children of $o/z_{d,j}^1/\cdots/z_{d,j}^{k-1}$, according to the multinomial $\theta_{d,o/z_{d,j}^1/\cdots/z_{d,j}^{k-1}}$. When a leaf node is reached, the word is generated from the multinomial distribution $\phi_{o/z_{d,j}^1/z_{d,j}^2/\cdots/z_{d,j}^h}$. Note that the length of the path h is not necessary to be equal for all documents, if not all leaf nodes are on the same level.

The whole generative process is illustrated in Figure 7.1. Table 7.1 collects the notations.

A feature supporting recursive construction. For every non-leaf topic node, we can derive a word distribution by marginalizing their child topic word distributions:

$$\phi_{t,x} = p(x|t) = \sum_{z=1}^{C_t} p(z|t)p(x|t/z) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z,x} \tag{7.1}$$

So in our model, the word distribution ϕ_t for an internal node in the topic hierarchy can be seen as a mixture of its child topic word distributions. The Dirichlet prior α_t determines the mixing weight.

A topical hierarchy $\mathcal{T}$ is parameterized by $\alpha_t(\mathcal{T})$ where $C_t(\mathcal{T}) > 0$, and $\phi_t(\mathcal{T})$ where $C_t(\mathcal{T}) = 0$. We define a topical hierarchy $\mathcal{T}_1$ to be *subsumed* by $\mathcal{T}_2$, if there is a mapping κ from node t in $\mathcal{T}_1$ to node t' in $\mathcal{T}_2$, such that for every node t in $\mathcal{T}_1$, $\pi_t(\mathcal{T}_1) = \pi_{\kappa(t)}(\mathcal{T}_2)$, and one of the following is true:

1. $C_t(\mathcal{T}_1) = C_{\kappa(t)}(\mathcal{T}_2) > 0$ and $\alpha_t(\mathcal{T}_1) = \alpha_{\kappa(t)}(\mathcal{T}_2)$; or

2. $C_t(\mathcal{T}_1) = 0$ and $\phi_t(\mathcal{T}_1) = \phi_{\kappa(t)}(\mathcal{T}_2)$.

In words, a subsumed tree is obtained by removing all the descendants of some nodes in a larger tree, and absorbing the word distributions of the descendants into the new leaf nodes. In Figure 7.2, we show three trees and each tree is subsumed by the one on its right. The subsumed tree retains equivalent high-level topic information of a larger tree, and can be recovered before we recover the larger tree. This idea allows us to recursively construct the whole hierarchy, which distinguishes our method from all-at-once construction methods. We present the recursive construction approach with justification in the next section.

7.3 The STROD algorithm

We propose a Scalable Tensor Recursive Orthogonal Decomposition (STROD) algorithm for topical hierarchy construction, the first that meets all the criteria in Section 7.1. It uses tensor (hypermatrix) decomposition to perform moment-based inference of the hierarchical topic model proposed in Section 7.2 recursively.

7.3.1 Moment-based inference

The central idea of the inference method is based on the *method of moments*, instead of *maximum likelihood*. It enables tractable computations to estimate the parameters.

In statistics, the *population moments* are expected values of powers of the random variable under consideration. The method of moments derives equations that relate the population moments to the model parameters. Then, it collects empirical population moments from observed samples, and solve the equations using the sample moments in place of the theoretical population moments. In our case, the random variable is the word occurring in each document. The population moments are expected occurrences and co-occurrences of the words. They are related to the model parameters α and ϕ. We can collect empirical population moments from the corpus, and estimate α and ϕ by fitting the empirical moments with theoretical moments. One particular computational advantage is that the inference only relies on the empirical population moments (word co-occurrence statistics).

They compress important information from the full data, and require only one scan of the data to collect.

The idea is promising, but not straightforward to apply to our model. The challenge is that the same population moments can be expressed by parameters on different levels. For the example in Figure 3.1, we can derive equations of the population moments (expected word co-occurrences) based on the model parameters associated with $t1, t2$ and $t3$, or based on those with $t4-t9$. Solving these equations independently will find 3 general topics and 6 more specific topics, but will neither reveal their relationship, nor guarantee the existence of the relationship.

Below we present a recursive inference method step by step and justify its correctness.

1. Conditional population moments. We consider the population moments *conditioned* on a non-leaf topic t. The first order moment is the expectation of word x's occurrence given that it is drawn from topic t's descendant. We have $p(x|t,\alpha) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z,x}$ according to Eq. (7.1).

The second order moment is the expectation of the co-occurrences of two words x_1 and x_2 given that they are both drawn from topic t's descendants.

$$\begin{aligned} p(x_1, x_2|t,t,\alpha) = &\sum_{z_1 \neq z_2} \frac{\alpha_{t,z_1}\alpha_{t,z_2}}{\alpha_{t,0}(\alpha_{t,0}+1)} \phi_{t/z_1,x_1}\phi_{t/z_2,x_2} \\ &+ \sum_{z=1}^{C_t} \frac{\alpha_{t,z}(\alpha_{t,z}+1)}{\alpha_{t,0}(\alpha_{t,0}+1)} \phi_{t/z,x_1}\phi_{t/z,x_2} \end{aligned} \tag{7.2}$$

Likewise, we can derive the third order moment as the expectation of co-occurrences of three words x_1, x_2 and x_3 given that they are all drawn from topic t's descendants:

$$\begin{aligned} &p(x_1, x_2, x_3|t,t,t,\alpha) \\ &= \sum_{z_1 \neq z_2 \neq z_3 \neq z_1} \frac{\alpha_{t,z_1}\alpha_{t,z_2}\alpha_{t,z_3}}{\alpha_{t,0}(\alpha_{t,0}+1)(\alpha_{t,0}+2)} \phi_{t/z_1,x_1}\phi_{t/z_2,x_2}\phi_{t/z_3,x_3} \\ &+ \sum_{z_1 \neq z_2} \frac{\alpha_{t,z_1}\alpha_{t,z_2}(\alpha_{t,z_1}+1)}{\alpha_{t,0}(\alpha_{t,0}+1)(\alpha_{t,0}+2)} (\phi_{t/z_1,x_1}\phi_{t/z_1,x_2}\phi_{t/z_2,x_3} \\ &+ \phi_{t/z_1,x_1}\phi_{t/z_1,x_3}\phi_{t/z_2,x_2} + \phi_{t/z_1,x_3}\phi_{z_1,x_2}\phi_{z_2,x_1}) \\ &+ \sum_{z=1}^{C_t} \frac{\alpha_{t,z}(\alpha_{t,z}+1)(\alpha_{t,z}+2)}{\alpha_{t,0}(\alpha_{t,0}+1)(\alpha_{t,0}+2)} \phi_{t/z,x_1}\phi_{t/z,x_2}\phi_{t/z,x_3} \end{aligned} \tag{7.3}$$

These equations exhibit good opportunities for a recursive solution, because the moments conditioned on a topic t can be expressed by only the Dirichlet prior and word distributions associated with its child topics. If these low order moments can uniquely determine the model parameters, we can use them to recover the child topics of every topic robustly, and by recursion, we can then construct the whole tree (Figure 7.2).

Fortunately, there is indeed a robust technique to recover the parameters from low order moments.

2. Tensor orthogonal decomposition. Anandkumar *et al.* [6] proved that with some mild non-degeneracy conditions, the following equations can be uniquely solved by a tensor orthogonal decomposition method:

$$M_2 = \sum_{z=1}^{k} \lambda_z v_z \otimes v_z, M_3 = \sum_{z=1}^{k} \lambda_z v_z \otimes v_z \otimes v_z \tag{7.4}$$

where M_2 is a $V \times V$ tensor (hence, a matrix) and M_3 is a $V \times V \times V$ tensor, λ_z is an unknown positive value about the weight of z-th component v_z, which is an unknown V-dimensional vector. In words, both M_2 and M_3 can be decomposed into the same number of components, and each component is determined by a single vector. The operator $\otimes$ denotes an outer product between tensors: if $A \in \mathbb{R}^{s_1 \times \cdots \times s_p}$, and $B \in \mathbb{R}^{s_{p+1} \times \cdots \times s_{p+q}}$, then $A \otimes B$ is a tensor in $\mathbb{R}^{s_1 \times \cdots \times s_{p+q}}$, and $[A \otimes B]_{i_1 \ldots i_{p+q}} = A_{i_1 \ldots i_p} B_{i_{p+1} \ldots i_{p+q}}$.

To write Eq. (7.1)-(7.3) in this form, we define:

$$M_1(t) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z} \tag{7.5}$$

$$E_2(t) = [p(x_1, x_2|t, t, \alpha)]_{V \times V} \tag{7.6}$$

$$M_2(t) = (\alpha_{t,0} + 1)E_2(t) - \alpha_{t,0} M_1(t) \otimes M_1(t) \tag{7.7}$$

$$E_3(t) = [p(x_1, x_2, x_3|t, t, t, \alpha)]_{V \times V \times V} \tag{7.8}$$

$$\begin{aligned} U_1(t) &= E_2(t) \otimes M_1(t), \\ &\quad U_2(t) = \Omega(U_1(t), 1, 3, 2), U_3(t) = \Omega(U_1(t), 2, 3, 1) \end{aligned} \tag{7.9}$$

$$\begin{aligned} M_3(t) &= \frac{(\alpha_{t,0}+1)(\alpha_{t,0}+2)}{2} E_3(t) + \alpha_{t,0}^2 M_1 \otimes M_1 \otimes M_1 \\ &\quad - \frac{\alpha_{t,0}(\alpha_{t,0}+1)}{2} [U_1(t) + U_2(t) + U_3(t)] \end{aligned} \tag{7.10}$$

where $\Omega(A, a, b, c)$ permutes the modes of tensor A, such that $\Omega(A, a, b, c)_{i_1,i_2,i_3} = A_{i_a,i_b,i_c}$. It follows that:

$$M_2(t) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z} \otimes \phi_{t/z}, M_3(t) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z} \otimes \phi_{t/z} \otimes \phi_{t/z}$$

So they fit Eq. (7.4) nicely, and intuitively. If we decompose $M_2(t)$ and $M_3(t)$, the z-th component is determined by the child topic word distribution $\phi_{t/z}$, and its weight is $\frac{\alpha_{t,z}}{\alpha_{t,0}}$, which is equal to $p(t/z|t, \alpha_t)$.

Algorithm 3 outlines the tensor orthogonal decomposition method for recovering the components. It can be partitioned into two parts:

1. Lines 3.1 to 3.4 project the large tensor $M_3 \in \mathbb{R}^{V \times V \times V}$ into a smaller tensor $\widetilde{T} \in \mathbb{R}^{k \times k \times k}$. $\widetilde{T}$ is not only of smaller size, but can be decomposed into an orthogonal form: $\widetilde{T} = \sum_{z=1}^{k} \widetilde{\lambda_i} \widetilde{v_i}^{\otimes 3}$. $\widetilde{v_i}, i = 1, \ldots, k$ are orthonormal vectors in $\mathbb{R}^k$. This is assured by the whitening matrix W calculated in Line 3.2, which satisfies $W^T M_2 W = I$.

2. Lines 3.5 to 3.14 perform orthogonal decomposition of $\widetilde{T}$ via a power iteration method. The orthonormal eigenpairs $(\widetilde{\lambda_z}, \widetilde{v_z})$ are found one by one. To find one such pair, the algorithm randomly starts with a unit-form vector v, runs power iteration (Line 3.9) for n times, and records the candidate eigenpair. This process further repeats by N times, starting from

Algorithm 3: Tensor Orthogonal Decomposition (TOD)

```
    Input: Tensor $M_2 \in \mathbb{R}^{V\times V}$, $M_3 \in \mathbb{R}^{V\times V\times V}$, number of components $k$, number of outer and inner
           iterations $N$ and $n$
    Output: The decomposed components $(\lambda_z, v_z), z = 1,\ldots,k$
3.1 Compute $k$ orthonormal eigenpairs $(\sigma_z, \mu_z)$ of $M_2$;
3.2 Compute a whitening matrix $W = M\Sigma^{-\frac{1}{2}}$ ; // $M = [\mu_1,\ldots,\mu_k], \Sigma = diag(\sigma_1,\ldots,\sigma_k), W^TM_2W = I$
3.3 Compute $(W^T)^+ = M\Sigma^{\frac{1}{2}}$ ;                // the Moore-Penrose pseudoinverse of $W^T$
3.4 Compute a $k^3$ tensor $\widetilde{T} = M_3(W,W,W)$ ;   // $\widetilde{T}_{i1,j1,k1} = \sum_{i2,j2,k2}(M_3)_{i2,j2,k2}W_{i2,i1}W_{j2,j1}W_{k2,k1}$
3.5 for $z = 1..k$ do
3.6     $\lambda^* \leftarrow 0$ ;                                 // the largest eigenvalue so far
3.7     for $outIter = 1..N$ do
3.8         $v \leftarrow$ a random unit-form vector;
3.9         for $innerIter = 1..n$ do  $v \leftarrow \frac{\widetilde{T}(I,v,v)}{||\widetilde{T}(I,v,v)||}$ ;          // power iteration update
3.10        if $\widetilde{T}(v,v,v) > \lambda^*$ then $(\lambda^*, v^*) \leftarrow (\widetilde{T}(v,v,v)$,v); // choose the largest eigenvalue
3.11    end
3.12    $\lambda_z = \frac{1}{(\lambda^*)^2}$, $v_z = \lambda_z(W^T)^+v^*$ ;        // recover eigenpair of the original tensor
3.13    $\widetilde{T} \leftarrow \widetilde{T} - \lambda^* v^* \otimes v^* \otimes v^*$ ;                               // deflation
3.14 end
3.15 return $(\lambda_z, v_z), z = 1,\ldots,k$
```

different unit-form vectors, and the candidate eigenpair with the largest eigenvalue is picked (Line 3.10). After an eigenpair is found, the tensor $\widetilde{T}$ is deflated by the found component (Line 3.13), and the same power iteration is applied to it to find the next eigenpair. After all the k orthonormal eigenpairs $(\widetilde{\lambda_z}, \widetilde{v_z})$ are found, they can be used to uniquely determine the k target components (λ_z, v_z) (Line 3.12).

The following theorem ensures that the decomposition is unique and fast.

Theorem 7.1 *Assume M_2 and M_3 are defined as in Eq. 7.4, $\lambda_z > 0$, and the vectors v_z's are linearly independent and have unit-form, then Algorithm 3 returns exactly the same set of (λ_z, v_z). Furthermore, the power iteration step of Line 3.9 converges in a quadratic rate.*

This theorem is essentially a combination of Theorem 4.3 and Lemma 5.1 in Anandkumar *et al.* [6].

Theorem 7.1 relies on several non-trivial claims: i) the orthogonal decomposition of $\widetilde{T}$ is unique; ii) the power iteration converges robustly and quickly to the eigenpair; and iii) a pair of $(\widetilde{\lambda_z}, \widetilde{v_z})$ uniquely determines a pair of (λ_z, v_z). The first two are proved in Anandkumar *et al.* [6], and the third can be proved by a similar proof of Theorem 4.1 in Anandkumar *et al.* [7]. We omit the details here. To see why these claims are non-trivial, we notice that the decomposition of

$M_2 = \sum_{z=1}^{k} \lambda_z v_z \otimes v_z$ is not unique. If (σ_z, μ_z) are orthonormal eigenpairs of M_2, then for any orthonormal matrix $O \in \mathbb{R}^{k\times k}$, $M_2 = \sum_{z=1}^{k} \sigma_z (O\mu_z) \otimes (O\mu_z)$. So there are infinite number of ways of decomposition if we only consider second order moments. This explains why CATHY's word co-occurrence network model has no robust inference method, since the word co-occurrence information is equivalent to the second order moments.

Anandkumar *et al.* [6] also provides perturbation analysis about Algorithm 3. When N and n are sufficiently large, the decomposition error is bounded by the error ϵ of empirical moments from theoretical moments. The number of required inner loop iterations n grows in a logarithm rate with k, and the outer loop N in a polynomial rate. They also proposed possible stopping criterion to reduce the number of trials of the random restart. Since the number of components k is bounded by a small constant $K \approx 10$ in our task, the power iteration update is very efficient, and we observe that $N = n = 30$ are sufficient.

The importance of Theorem 7.1 is that it allows us to use moments only up to the third order to recover the exact components, and the convergence is fast.

3. Recursive decomposition. With Algorithm 3 as a building block, we can divide and conquer the inference of the whole model. We devise Algorithm 4, which recursively infers model parameters in a top-down manner. Taking any topic node t as input, it computes the conditional moments $M_2(t)$ and $M_3(t)$. If t is not root, they are computed from the parent topic's moments and estimated model parameters. For example, according to Bayes's theorem,

$$\begin{aligned}
&[E_2(t)]_{x_1,x_2} = p(x_1, x_2|t, t, \alpha) \propto p(x_1, x_2, t, t|\pi_t, \pi_t, \alpha) \\
&= p(x_1, x_2|\pi_t, \pi_t, \alpha) p(t, t|x_1, x_2, \pi_t, \pi_t, \alpha) \\
&= [E_2(\pi_t)]_{x_1,x_2} \alpha_{\pi_t,\chi_t} (\alpha_{\chi_t,z} + 1) \phi_{t,x_1} \phi_{t,x_2} \\
&/ \left(\sum_{z=1}^{C_{\pi_t}} \alpha_{\pi_t,z} (\alpha_{\pi_t,z} + 1) \phi_{\pi_t/z,x_1} \phi_{\pi_t/z,x_2} + \sum_{z_1 \neq z_2} \alpha_{\pi_t,z_1} \alpha_{\pi_t,z_2} \phi_{\pi_t/z_1,x_1} \phi_{\pi_t/z_2,x_2} \right)
\end{aligned} \tag{7.11}$$

Other quantities in Eq. (7.5)-(7.10) can be computed similarly. Then it performs tensor decomposition and recovers the parameter α_t and $\phi_{t/z}$ for each child topic. It then enumerates its children and makes recursive calls with each of them as input. The recursion stops when reaching leaf nodes, where $C_t = 0$. A call of Algorithm 4 with the root o as input will construct the entire hierarchy.

```
Algorithm 4: Recursive Tensor Orthogonal Decomposition (RTOD)
    Input: topic $t$, number of outer and inner iterations $N, n$
4.1 Compute $M_2(t)$ and $M_3(t)$ ;                // only relies on $t$'s ancestors
4.2 $(\lambda_z, v_z) \leftarrow TOD(M_2(t), M_3(t), C_t, N, n)$;
4.3 $\alpha_{t,z} = \alpha_{t,0}\lambda_z, \phi_{t/z} = v_z$;
4.4 for $z = 1..C_t$ do
4.5     RTOD($t/z, N, n$) ;   // Recursion, get the parameters for each subtree
4.6 end
```

The correctness of this recursive algorithm is permitted by the special structure of our model. In particular, we have the following theorem.

Theorem 7.2 *The RTOD algorithm (Algorithm 4) has both robust recovery and robust revision property.*

The robust recovery property follows Theorem 7.1, plus the fact that the conditional moments of a topic can be expressed by only the Dirichlet prior and word distributions associated with its child topics. The robust revision property is due to conditional independence during the recursive construction procedure: i) once a topic t has been visited in the algorithm, the construction of its children is independent of each other; and ii) the conditional moments $M_2(t)$ and $M_3(t)$ can be computed independently of t's descendants. In fact, it leads to a stronger claim.

Corollary 7.1 *If $\mathcal{T}_1$ is subsumed by $\mathcal{T}_2$ with the mapping $\kappa(\cdot)$, then the RTOD algorithm on $\mathcal{T}_1$ and $\mathcal{T}_2$ returns identical parameters for $\mathcal{T}_1$ and $\kappa(\mathcal{T}_1)$.*

Therefore, the tree topology can be expanded or varied locally with minimal revision to the inferred topics. This is in particular useful when the structure of the topic tree is not fully determined in the beginning. The recursive construction offers users a chance to see the construction results and interact with the topic tree expansion or its local variations by deciding on the number of topics.

7.3.2 Scalability improvement

Although Algorithms 3 and 4 are robust, they are not scalable. The orthogonal decomposition of the tensor $\widetilde{T} \in \mathbb{R}^{k\times k\times k}$ (Lines 3.5-3.14) is efficient, because k is small. However, the bottleneck of the computation is preparing the tensor $\widetilde{T}$, including Line 4.1 and Lines 3.1 to 3.4. They involve the creation of a dense tensor $M_3 \in \mathbb{R}^{V\times V\times V}$, and the time-consuming operation $M_3(W, W, W)$. Since V is usually tens of thousands or larger, it is impossible to store such a tensor in memory and perform the tensor product operation. In fact, even the second order moment $M_2 \in \mathbb{R}^{V\times V}$ is dense and large, challenging both space and time efficiency already.

Anandkumar [6] discusses a plausible way to solve the space challenge, by avoiding explicit creation of the tensors M_3 and $\widetilde{T}$. It suggests going through the document-word occurrence data for computing the power iterations. This requires as many times of data passes as other inference methods, if not more. Therefore, its scalability will still be unsatisfactory.

We make key contributions to solving the challenge in a different approach. We avoid explicit creation of both tensor M_3 and M_2. But we do explicitly create $\widetilde{T}$ since it is memory efficient. Therefore, the efficient power iteration updates remain as in Algorithm 3. Utilizing the special structure of the tensors in our problem, we show that $\widetilde{T}$ can be created by passing the data only twice, without incurring creations of any dense V^2 or V^3 tensors.

1. Avoid creating $\mathbf{M_2}$. For ease of discussion, we omit the conditional topic t in the notation of this and next subsection. According to Eq. (7.7), $M_2 = (\alpha_0 + 1)E_2 - \alpha_0 M_1 \otimes M_1$. E_2 is a sparse symmetric matrix because only two words co-occurring in one document will contribute to the empirical estimation of E_2. However, $M_1 \otimes M_1$ is a full V by V matrix. We would like to compute the whitening matrix W without explicit creation of M_2.

First, we notice that M_1 is in the *column space* of M_2 (*i.e.*, M_1 is a linear combination of M_2's column vectors), so E_2 has the same column space S as M_2. Also, since $M_2 = \sum_{z=1}^{k} \lambda_z v_z \otimes v_z$ is positive definite, so is $E_2 = \frac{1}{\alpha_0+1}(M_2 + \alpha_0 M_1 \otimes M_1)$. Let $E_2 = U\Sigma_1 U^T$ be its spectral decomposition, where $U \in \mathbb{R}^{V\times k}$ is the matrix of k eigenvectors, and $\Sigma_1 \in \mathbb{R}^{k\times k}$ is the diagonal eigenvalue matrix. The k column vectors of U form an orthonormal basis of S. M_1's representation in this basis is $M_1' = U^T M_1$. Now, M_2 can be written as:

$$M_2 = U[(\alpha_0 + 1)\Sigma_1 - \alpha_0 M_1' \otimes M_1']U^T$$

So, a second spectral decomposition can be performed on $M_2' = (\alpha_0 + 1)\Sigma_1 - \alpha_0 M_1' \otimes M_1'$, as

$M_2' = U'\Sigma U'^T$. Then we have:

$$M_2 = UU'\Sigma(UU')^T$$

Therefore, we effectively obtain the spectral decomposition of $M_2 = M\Sigma M^T$ without creating M_2. Not only the space requirement is reduced (from a dense $V \times V$ matrix to a sparse matrix E_2), but also the time for spectral decomposition. If we perform spectral decomposition for M_2 directly, it requires $\mathcal{O}(V^3)$ time complexity. However, using the twice spectral decomposition trick above, we just need to compute the first largest k eigenpairs for a sparse matrix E_2, and a spectral decomposition for a small matrix $M_2' \in \mathbb{R}^{k\times k}$. The first decomposition can be done efficiently by a power iteration method or other more advanced algorithms. The time complexity is roughly $\mathcal{O}(k\|E_2\|_0)$, where $\|E_2\|_0$ is the number of non-zero elements in E_2. The second decomposition requires $\mathcal{O}(k^3)$ time, which can be regarded as constant since $k <= K \approx 10$.

2. Avoid creating $\mathbf{M_3}$. The idea is to directly compute $\widetilde{T} = M_3(W, W, W)$ without creating M_3. This is possible due to the distributive law: $(A + B)(W, W, W) = A(W, W, W) + B(W, W, W)$. The key is to decouple M_3 as a summation of many different tensors, such that the computation of the product between each tensor and W is easy.

We begin with the empirical estimation of E_3. Suppose we use $c_{i,x}$ to represent the count of word x in document d_i. Then E_3 can be estimated by averaging all the 3-word triples in each document:

$$\begin{aligned}
E_3 &= \frac{1}{D}\left[A_1 - A_2 - \Omega(A_2, 2, 1, 3) - \Omega(A_2, 2, 3, 1) + 2A_3\right] \\
A_1 &= \sum_{i=1}^{D} \frac{1}{l_i(l_i-1)(l_i-2)} c_i \otimes c_i \otimes c_i \\
A_2 &= \sum_{i=1}^{D} \frac{1}{l_i(l_i-1)(l_i-2)} c_i \otimes diag(c_i) \\
A_3 &= \sum_{i=1}^{D} \frac{1}{l_i(l_i-1)(l_i-2)} tridiag(c_i)
\end{aligned} \tag{7.12}$$

where $tridiag(v)$ is a tensor with vector v on its diagonal: $tridiag(v)_{i,i,i} = v_i$. Let $s_i = \frac{1}{l_i(l_i-1)(l_i-2)}$.

From the fact $(v \otimes v \otimes v)(W, W, W) = (W^T v) \otimes (W^T v) \otimes (W^T v) = (W^T v)^{\otimes 3}$, we can derive:

$$A_1(W, W, W) = \sum_{i=1}^{D} s_i (W^T c_i)^{\otimes 3} \tag{7.13}$$

Based on another fact, $(v \otimes M)(W, W, W) = (W^T v) \otimes M(W, W) = (W^T v) \otimes W^T M W$, we can derive:

$$A_2(W, W, W) = \sum_{i=1}^{D} s_i (W^T c_i) \otimes W^T diag(c_i) W \tag{7.14}$$

Let W_x^T be the x-th column of W^T, we have:

$$A_3(W, W, W) = \sum_{x=1}^{V} \sum_{i=1}^{D} s_i c_{i,x} (W_x^T)^{\otimes 3} \tag{7.15}$$

So we do not need to explicitly create E_3 to compute $E_3(W, W, W)$. The time complexity using Eq. (7.13)-(7.15) is $\mathcal{O}(Lk^2)$, which is equivalent to $\mathcal{O}(L)$ because k is small.

Using the same trick, we can obtain:

$$U_1(W, W, W) = W^T E_2 W \otimes W^T M_1 \tag{7.16}$$

$$(M1 \otimes M1 \otimes M1)(W, W, W) = (W^T M_1)^{\otimes 3} \tag{7.17}$$

Eq. (7.16) requires $\mathcal{O}(k^2 \|E_2\|_0)$ time to compute, while $\|E_2\|_0$ can be large. We can further speed it up.

We notice that $W^T M_2 W = I$ by definition. Substituting M_2 with Eq. (7.7), we have:

$$W^T[(\alpha_0 + 1)E_2 - \alpha_0 M_1 \otimes M_1]W = I \tag{7.18}$$

which is followed by:

$$W^T E_2 W = \frac{1}{(\alpha_0 + 1)}[I + \alpha_0 (W^T M_1)^{\otimes 2}] \tag{7.19}$$

Pluging Eq. (7.19) into (7.16) further reduces the complexity of computing $U_1(W, W, W)$ to $\mathcal{O}(Vk+k^3)$. $U_2(W, W, W)$ and $U_3(W, W, W)$ can be obtained by permuting $U_1(W, W, W)$'s modes, in $\mathcal{O}(k^3)$

time.

Putting these together, we have the following fast computation of $\widetilde{T} = M_3(W, W, W)$ by passing the data once:

$$\begin{aligned} \widetilde{T} = M_3(W, W, W) = \frac{(\alpha_{t,0} + 1)(\alpha_{t,0} + 2)}{2} E_3(W, W, W) \\ -\frac{\alpha_{t,0}(\alpha_{t,0} + 1)}{2} [(U_1 + U_2 + U_3)(W, W, W)] + \alpha_{t,0}^2 (W^T M_1)^{\otimes 3} \end{aligned} \tag{7.20}$$

which requires $\mathcal{O}(Lk^2 + Vk^2 + k^3) = \mathcal{O}(L)$ time in total.

3. Estimation of empirical conditional moments. To estimate the empirical conditional moments for topic t, we estimate the 'topical' count of word x in document d_i as:

$$c_{i,x}(t) = c_{i,x} p(t|x) = c_{i,x}(\pi_t) \frac{\alpha_{\pi_t, \chi_t} \phi_{t,x}}{\sum_{z=1}^{C_{\pi_t}} \alpha_{\pi_t, z} \phi_{\pi_t/z, x}} \tag{7.21}$$

and $c_{i,x}(o) = c_{i,x}$. Then we can estimate M_1 and E_2 using these empirical counts:

$$\begin{aligned} M_1(t) &= \sum_{i=1}^{D} \frac{1}{l_i(t)} c_i(t) \\ E_2(t) &= \sum_{i=1}^{D} \frac{1}{l_i(t)(l_i(t) - 1)} [c_i(t) \otimes c_i(t) - diag(c_i(t))] \end{aligned} \tag{7.22}$$

where $l_i(t) = \sum_{x=1}^{V} c_{i,x}(t)$. These enable fast estimation of empirical moments by passing data once.

Finally, we have a scalable tensor recursive orthogonal decomposition algorithm as outlined in Algorithm 5.

Due to the exact equivalence of the computation procedures, we have the following theorem.

Theorem 7.3 *STROD (Algorithm 5) has both robust recovery and robust revision property.*

We notice that the hyperparameter $\alpha_{t,0}$ and the number of child topics C_t are needed to run the STROD algorithm. We discuss how to learn them automatically in next subsection.

Algorithm 5: Scalable Tensor Recursive Orthogonal Decomposition (STROD)

```
     Input: topic $t$, number of outer and inner iterations $N, n$
5.1  Compute $M_1(t)$ and $E_2(t)$ according to Eq. (7.22);
5.2  Find $k$ largest orthonormal eigenpairs $(\sigma_z, \mu_z)$ of $E_2$;
5.3  $M'_1 = UM_1(t)$ ;                              // $U = [\mu_1, \dots, \mu_k], \Sigma_1 = diag(\sigma_1, \dots, \sigma_k)$
5.4  Compute spectral decomposition for $M'_2 = (\alpha_{t,0} + 1)\Sigma_1 - \alpha_{t,0} M'_1 \otimes M'_1 = U'\Sigma U'^T$;
5.5  $M = UU', W = M\Sigma^{-\frac{1}{2}}, (W^T)^+ = M\Sigma^{\frac{1}{2}}$;
5.6  Compute $\widetilde{T} = M_3(W, W, W)$ according to Eq. (7.20);
5.7  Perform power iteration Line 3.5 to 3.14 in Algorithm 3;
5.8  $\alpha_{t,z} = \alpha_{t,0}\lambda_z, \phi_{t/z} = v_z$;
5.9  for $z = 1..C_t$ do
5.10     STROD$(t/z, N, n)$ ;
5.11 end
```

7.3.3 Hyperparameter learning

1. Selection of the number of topics. We discuss how to select C_t when the tree width K is given. We first compute the largest K eigenvalues of E_2 in Line 5.2, and then select the smallest k such that the first k eigenpairs form a subspace that is good approximation of E_2's column space. This is similar to the idea of using Principle Component Analysis (PCA) to select a small subset of the eigenvectors as basis vectors. The *cumulative energy* $g(k)$ for the first k eigenvectors is defined to be $g(k) = \sum_{z=1}^{k} \sigma_z$. And we choose the smallest value of k such that $\frac{g(k)}{g(K)} > \eta$, and let $C_t = k$. $\eta \in [0, 1]$ controls the required energy of the first k eigenvectors, and can be tuned according to the application. When $\eta = 1$ a full K-branch tree will be constructed. When $\eta = 0$ the tree contains a single root node because $C_o = 0$. Typically η between 0.7 and 0.9 results in reasonable children numbers.

2. Learning Dirichlet prior. First, we note that the individual prior $\alpha_{t,z}$ can be learned by the decomposition algorithm, when the summation $\alpha_{t,0}$ of $\alpha_{t,1}$ to α_{t,C_t} is given to perform the inference for topic t. This already largely reduces the number of hyperparameters that are needed to be given. Large $\alpha_{t,0}$ indicates that t's subtopics tend to be mixed together in a document, while small $\alpha_{t,0}$ suggests that a document usually talks about only a few of the subtopics. When $\alpha_{t,0}$ approaches 0, one expects a document to have only one subtopic of t. So $\alpha_{t,0}$ can usually be set empirically according to the prior knowledge of the documents, such as 1 to 100.

If one wants to learn $\alpha_{t,0}$ automatically, we propose a heuristic method hereby. Suppose the data are generated by an authentic $\alpha^*_{t,0}$, and the moments are computed using the same $\alpha^*_{t,0}$, then the decomposition result should satisfy $\sum_{z=1}^{C_t} \alpha_{t,z} = \alpha_{t,0}$ exactly. However, if one uses a different $\alpha_{t,0}$ to compute the moments, the moments could deviate from the true value and result in mismatched $\alpha_{t,z}$. The discrepancy between returned $\sum_{z=1}^{C_t} \alpha_{t,z}$ and initial $\alpha_{t,0}$ indicate how much $\alpha_{t,0}$ deviates from the authentic value. So we can use the following fixed-point method to learn $\alpha_{t,0}$, where δ is learning rate.

1. Initialize $\alpha_{t,0} = 1$;
2. While (not converged)
   (a) Perform tensor decomposition for topic t to update $\alpha_{t,z}, z = 1, \ldots, C_t$;
   (b) $\alpha'_{t,0} = \sum_{z=1}^{C_t} \alpha_{t,z}$;
   (c) Update $\alpha_{t,0} \leftarrow \alpha_{t,0} + \delta(\alpha'_{t,0} - \alpha_{t,0})$;

7.4 Experiments

In this section we first introduce the datasets and the methods used for comparison, and then evaluate on *scalability*, *robustness*, and *interpretability*.

Datasets. Our performance study is on four datasets:

- DBLP title: A set of titles of recently published papers in DBLP[2]. The set has 1.9M titles, 152K unique words, and 11M tokens.
- CS abstract: A dataset of computer science paper abstracts from Arnetminer[3]. The set has 529K papers, 186K unique words, and 39M tokens.
- TREC AP news: A TREC news dataset (1998). It contains 106K full articles, 170K unique words, and 19M tokens.
- Pubmed abstract: A dataset of life sciences and biomedical topic. We crawled 1.5M abstracts[4] from Jan. 2012 to Sep. 2013. The dataset has 98K unique words after stemming and 169M tokens.

[2]http://www.dblp.org
[3]http://www.arnetminer.org
[4]http://www.ncbi.nlm.nih.gov/pubmed

We remove English stopwords from all the documents. Documents shorter than 3 tokens are not used for computing the moments because we rely on up to third order moments.

Methods for comparison. Our method is compared with the following topical hierarchy construction methods.

- hPAM – parametric hierarchical topic model. The hierarchical Pachinko Allocation Model [61] is a state-of-the-art parametric hierarchical topic modeling approach. hPAM outputs a specified number of supertopics and subtopics, as well as the associations between them.
- nCRP – nonparametric hierarchical topic model. Although more recently published nonparametric models have more capability in document modeling, their scalability is worse than nested Chinese Restaurant Process [32]. So we choose nCRP to represent this category. It outputs a tree with a specified height, but the number of topics is determined by the algorithm. A hyperparameter can be tuned to generate more or fewer topics. In our experiment we tune it to generate an approximately identical number of topics as other methods.
- splitLDA – recursively applying LDA. We implement a recursive method described by Pujara and Skomoroch [70]. They use LDA to infer topics for each level, and split the corpus according to the inferred results to produce a smaller corpus for inference with the next level. This heuristic method has the best known scalability so far. For fair comparison of the fundamental computational complexity of different algorithms, we do not use any parallel implementation for all the methods. So we implement splitLDA on top of a fast single-machine LDA inference algorithm [98].
- CATHY – recursively clustering word co-occurrence networks. The method we introduced in Section 3.1.
- STROD – and its variations RTOD, RTOD$_2$, RTOD$_3$. This is our scalable tensor recursive orthogonal decomposition method. We implement several versions to analyze our scalability improvement techniques: (i) RTOD: recursive tensor orthogonal decomposition without scalability improvement (Algorithm 4); (ii) RTOD$_2$: RTOD plus the technique of avoiding creation of M_2; (iii) RTOD$_3$: RTOD plus the technique of avoiding creation of M_3; and (iv) STROD: Algorithm 5 with the full scale-up technique.

We use an optimized Java implementation MALLET [59] for the first three Gibbs sampling-based methods, and set the number of iterations to be 1000, which is the common practice. We implement CATHY and STROD in MATLAB because Java does not have good support with matrix computation and spectral algorithms.

7.4.1 Scalability

The first evaluation assesses the scalability of different algorithms, which is one focal point of this chapter.

Figure 7.3 shows the overall runtime in these datasets. STROD is several orders of magnitude faster than the existing methods. On the largest dataset it reduces the runtime from one or more days to 18 minutes. CATHY is the second best method in short documents such as titles and abstracts because it compresses the documents into word co-occurrence networks. But it is still more than 100 times slower than STROD due to many rounds of EM iterations. splitLDA and hPAM rely on Gibbs sampling, and the former is faster because it recursively performs LDA, and considers fewer dependencies in sampling. nCRP is two orders of magnitude slower due to its nonparametric nature.

We then conduct analytical study of the runtime growth with respect to different factors. Figures 7.4-7.6 show the runtime varying with the number of tokens, the tree height and the tree width. We can see that the runtime of STROD grows slowly, and it has the best performance in all occasions. In Figure 7.5, we exclude hPAM because it is designed for $H = 2$. In Figure 7.6, we use the same number of child topics C_t for each node for all the methods. We exclude nCRP from all these experiments because it takes too long time to finish (>90 hours with 600K tokens).

Figure 7.7 shows the performance in comparison with the slower variations of STROD. Both RTOD and RTOD$_2$ fail to finish when the vocabulary size grows beyond 1K, because the third-order moment tensor M_3 requires $\mathcal{O}(V^3)$ space to create. RTOD$_3$ also has limited scalability because the second order moment tensor $M_2 \in \mathbb{R}^{V \times V}$ is dense. STROD scales up easily by avoiding explicit creation of these tensors.

7.4.2 Robustness

The second evaluation assesses the robustness of different algorithms. For each dataset, we sample 10, 000 documents and run each algorithm 10 times and measure the *variance* among the 10 runs for the same method as follows. Each pair of algorithm runs generate the same number of topics, but their correspondence is unknown. For example, the topic $o/1$ in the first run may be close to $o/3$ in the second run. We measure the KL divergence between all pairs of topics between the two runs, build a bipartite graph using the negative KL divergence as the edge weight, and then use a maximum matching algorithm to determine the best correspondence (top-down recursively). Then we average the KL divergence between matched pairs as the difference between the two algorithm runs. Finally, we average the difference between all $10 \times 9 = 90$ ordered pairs of algorithm runs as the final variance. We exclude nCRP in this section, since even the number of topics is not a constant after each run. Due to space limitation, we report the variance on the first three datasets.

Table 7.2 summarizes the results: STROD has lowest variance in all the three datasets. The other three methods based on Gibbs sampling have variance larger than 1 in all datasets, which implies that the topics generated across multiple algorithm runs are considerably different.

We also evaluate the variance of STROD when we vary the number of outer and inner iterations N and n. As shown in Figure 7.8, the variance of STROD quickly diminishes when the number of outer and inner iterations grow to 10. The same trend is true for other datasets. This validates the theoretical analysis of their fast convergence and the guarantee of robustness.

In conclusion, STROD achieves robust performance with small runtime. It is stable and reliable to be used as a hierarchy construction method for large text collections.

7.4.3 Interpretability

The final evaluation assesses the interpretability of the constructed topical hierarchy, via human judgment. We evaluate hierarchies constructed from DBLP titles and TREC AP news. For simplicity, we set the number of subtopics to be 5 for all topics. For hPAM, we post-process them to obtain the 5 strongest subtopics for each topic. For all the methods we use the same phrase mining and ranking procedure to enhance the interpretability. We do not include nCRP in this study because hPAM has been shown to have superior performance of it [61].

In order to evaluate the topic coherence and parent-child relationship, we use two *intrusion detection* tasks which were introduced in Section 3.3.2.

- Phrase Intrusion (PI): X phrases are shown to an evaluator. One is a top-10 phrase from a sibling topic, and the remaining ones come from the top-10 phrases of the same topic. Evaluators are asked to select the intruder phrase, or to indicate that they are unable to make a choice.
- Topic Intrusion (TI): Evaluators are shown a parent topic t and X candidate child topics. $X-1$ of the child topics are actual children of t in the generated hierarchy, and the remaining child topic is not. Each topic is represented by its top-5 ranked phrases. Evaluators are asked to select the intruder child topic, or to indicate that they are unable to make a choice.

For this study we set $X = 4$. 160 Topic Intrusion questions and 200 Phrase Intrusion questions are randomly generated from the hierarchies constructed by these methods. We then calculate the agreement of the human choices with the actual hierarchical structure constructed by the various methods. We consider a higher match between a given hierarchy and human judgment to imply a higher quality hierarchy. For each method, we report the F1 measure of the questions answered 'correctly' (matching the method) and consistently by three human judgers with CS background.

Figure 7.9 summarizes the results. STROD is among the best performing methods in both tasks. This suggests that the quality of the hierarchy is not compromised by the strong scalability and robustness of STROD. As expected, splitLDA and STROD perform similarly in PI task, since they share the same LDA process for one level. However, STROD has a more principled model and theoretically guaranteed inference method to construct the hierarchy. That leads to more meaningful parent-child relations, and thus better performace in TI task.

A subset of the hierarchy constructed from CS abstract is presented in Figure 7.10. For each non-root node, we show the top ranked phrases. Node o/1 is about 'data', while its children involves database, data mining and bioinformatics. The lower the level is, the more pure the topic is, and the more multigrams emerge ahead of unigrams in general.

Figures and tables

Table 7.1: Notations used in our model

Symbol	Description
D	the number of documents in the corpus
V	the number of unique words in the corpus
$w_{d,j}$	the j-th word in the d-th document
l_d	the length (number of tokens) of document d
L	the total number of tokens in the corpus $\sum_{d=1}^{D} l_d$
π_t	the parent topic of topic t
χ_t	the suffix of topic t's notation ($t = \pi_t/\chi_t$)
C_t	the number of child topics of topic t
o	the root topic
ϕ_t	the multinomial distribution over words in topic t
α_t	the Dirichlet hyperparameter vector of topic t
$\theta_{d,t}$	the distribution over child topics of t for document d
T	the total number of topics in the hierarchy
τ	the number of leaf topics in the hierarchy

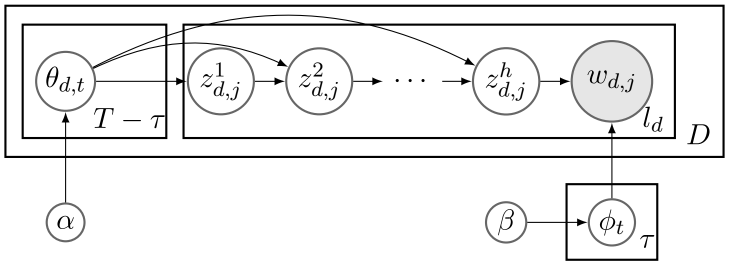


Figure 7.1: Latent Dirichlet allocation with topic tree

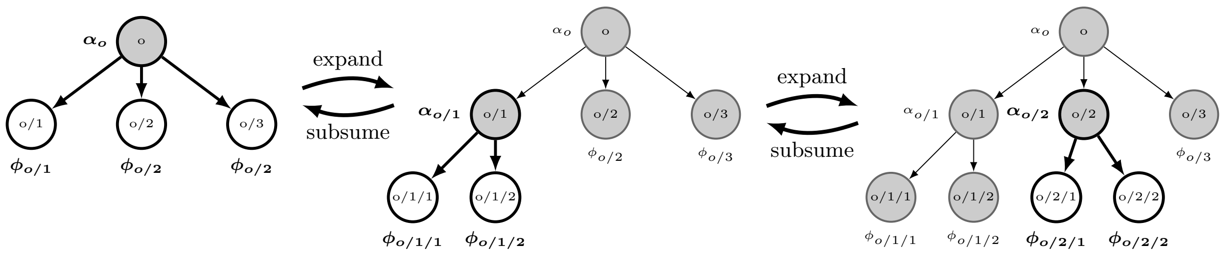


Figure 7.2: An illustration of recursive topical hierarchy construction. The construction order is from left to right. Each time one leaf topic node is expanded into several child topics (unshaded) and the relevant parameters (in bold) are estimated. The same figure explains *subsumption* relationship: A tree on the left is subsumed by a tree on the right

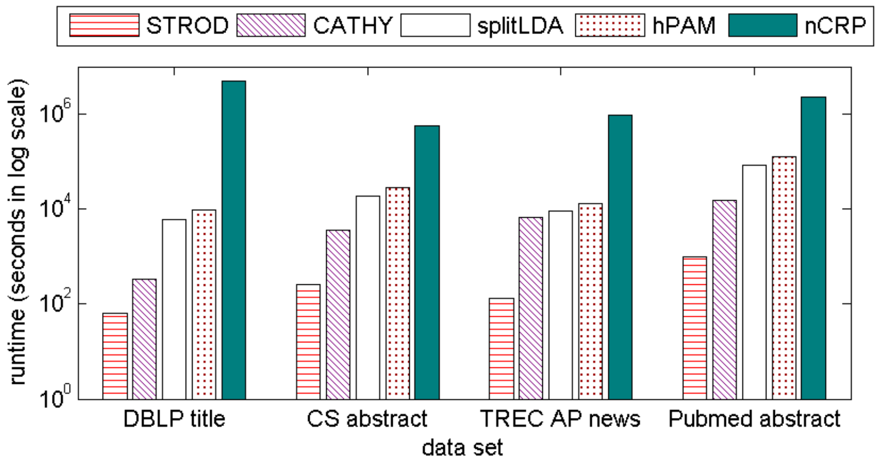


Figure 7.3: Total runtime on each dataset, $h = 2, C_t = 5$

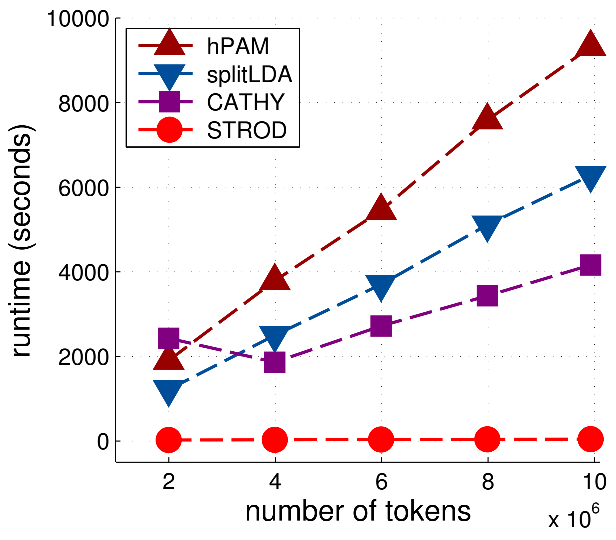


Figure 7.4: Runtime w.r.t corpus size

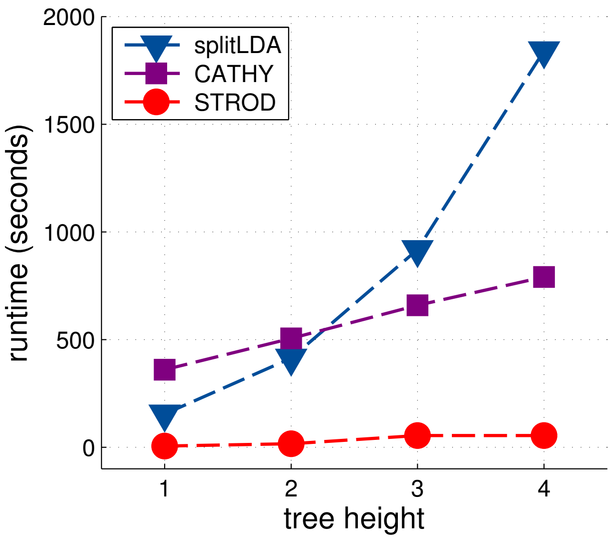


Figure 7.5: Runtime w.r.t tree height

Table 7.2: The variance of multiple algorithm runs in each dataset

Method	**DBLP title**	**CS abstract**	**TREC AP news**
hPAM	5.578	5.715	5.890
splitLDA	3.393	1.600	1.578
CATHY	17.34	1.956	1.418
STROD	**0.6114**	**0.0001384**	**0.004522**

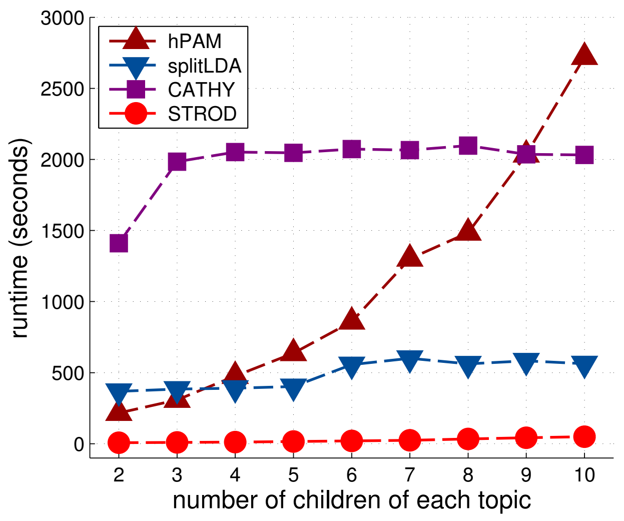


Figure 7.6: Runtime w.r.t the number of children for each topic

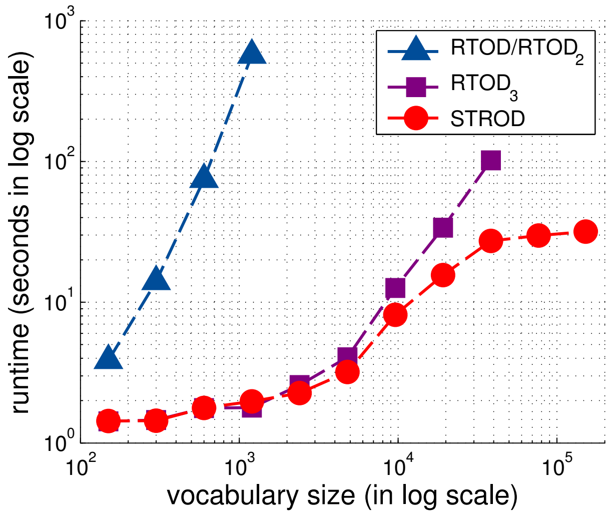
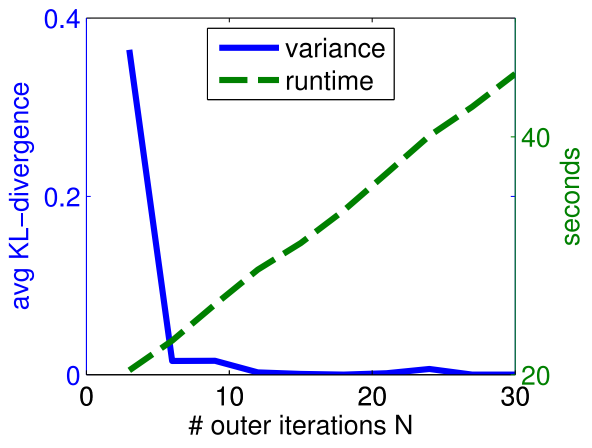
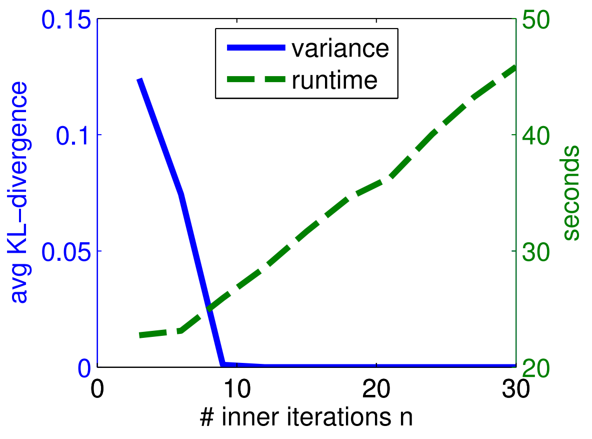


Figure 7.7: STROD scales much better than its variations (all except STROD fail to scale beyond 50K vocabulary size due to memory constraints)



(a) varying N when $n = 30$



(b) varying n when $N = 30$

Figure 7.8: The variance and runtime of STROD when varying # outer and inner iterations N and n (CS abstract)

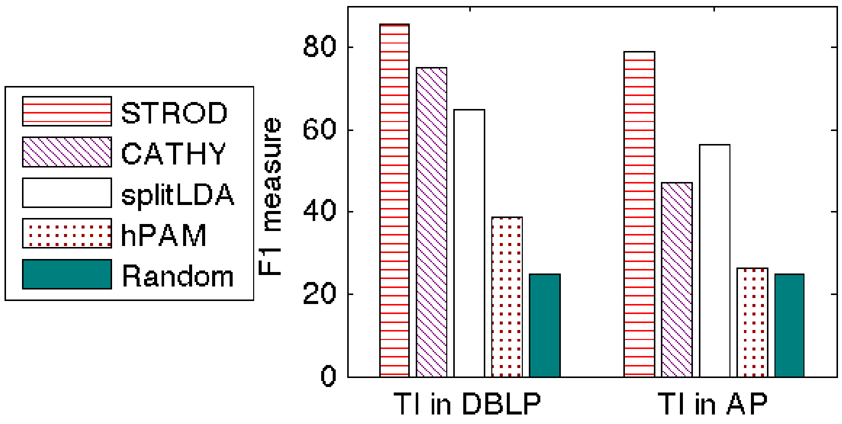


Figure 7.9: Phrase intrustion and topic intrustion study

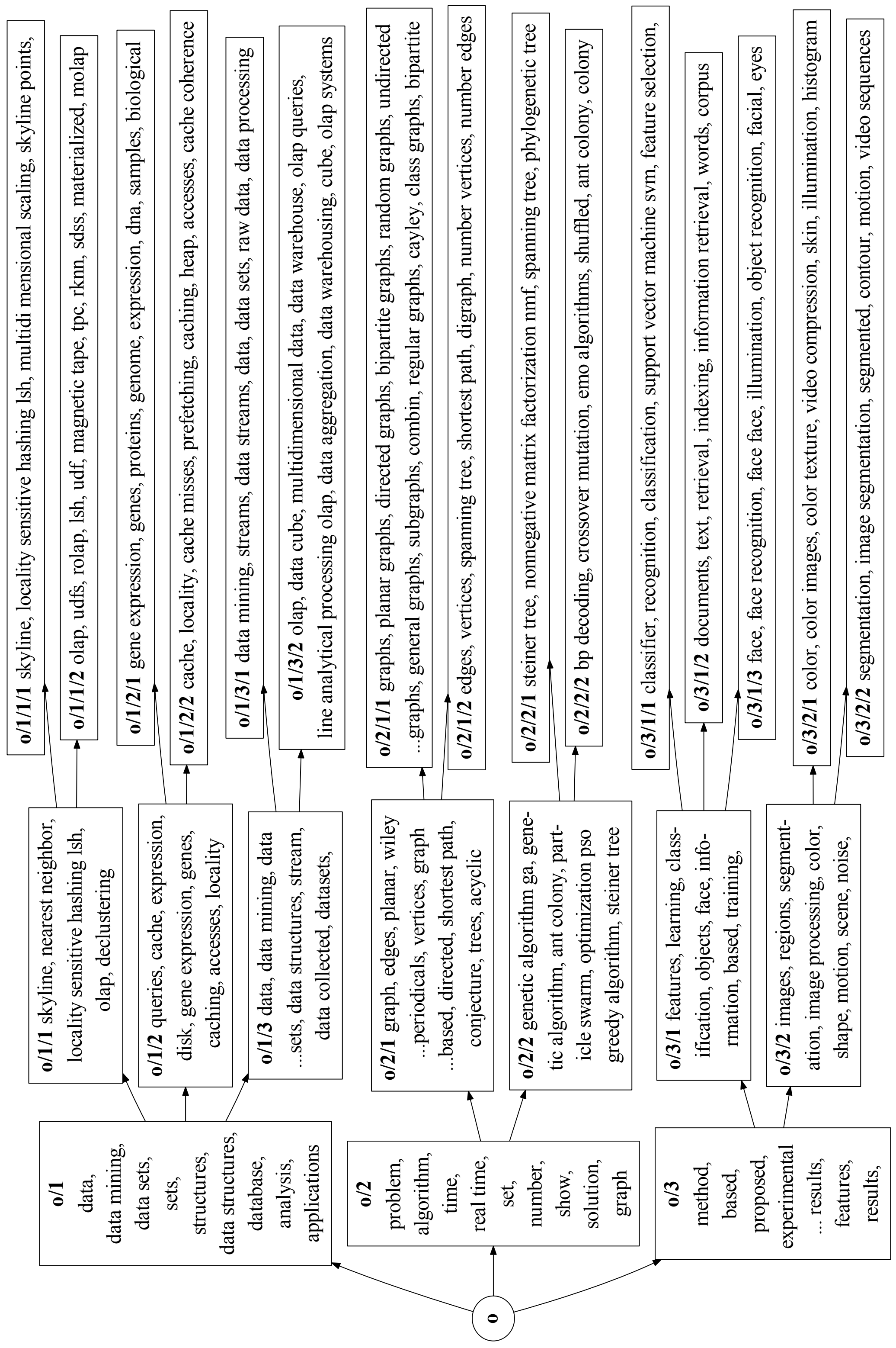

o
o/1 data, data mining, data sets, sets, structures, data structures, database, analysis, applications
o/2 problem, algorithm, time, real time, set, number, show, solution, graph
o/3 method, based, proposed, experimental ... results, features, results,
o/1/1 skyline, nearest neighbor, locality sensitive hashing lsh, olap, declustering
o/1/2 queries, cache, expression, disk, gene expression, genes, caching, accesses, locality
o/1/3 data, data mining, data ...sets, data structures, stream, data collected, datasets,
o/2/1 graph, edges, planar, wiley ...periodicals, vertices, graph ...based, directed, shortest path, conjecture, trees, acyclic
o/2/2 genetic algorithm ga, genetic algorithm, ant colony, particle swarm, optimization pso greedy algorithm, steiner tree
o/3/1 features, learning, classification, objects, face, information, based, training,
o/3/2 images, regions, segmentation, image processing, color, shape, motion, scene, noise,
o/1/1/1 skyline, locality sensitive hashing lsh, multidi mensional scaling, skyline points,
o/1/1/2 olap, udfs, rolap, lsh, udf, magnetic tape, tpc, rknn, sdss, materialized, molap
o/1/2/1 gene expression, genes, proteins, genome, expression, dna, samples, biological
o/1/2/2 cache, locality, cache misses, prefetching, caching, heap, accesses, cache coherence
o/1/3/1 data mining, streams, data streams, data, data sets, raw data, data processing
o/1/3/2 olap, data cube, multidimensional data, data warehouse, olap queries, line analytical processing olap, data aggregation, data warehousing, cube, olap systems
o/2/1/1 graphs, planar graphs, directed graphs, bipartite graphs, random graphs, undirected ...graphs, general graphs, subgraphs, combin, regular graphs, cayley, class graphs, bipartite
o/2/1/2 edges, vertices, spanning tree, shortest path, digraph, number vertices, number edges
o/2/2/1 steiner tree, nonnegative matrix factorization nmf, spanning tree, phylogenetic tree
o/2/2/2 bp decoding, crossover mutation, emo algorithms, shuffled, ant colony, colony
o/3/1/1 classifier, recognition, classification, support vector machine svm, feature selection,
o/3/1/2 documents, text, retrieval, indexing, information retrieval, words, corpus
o/3/1/3 face, face recognition, face face, illumination, object recognition, facial, eyes
o/3/2/1 color, color images, color texture, video compression, skin, illumination, histogram
o/3/2/2 segmentation, image segmentation, segmented, contour, motion, video sequences

Figure 7.10: Sample of hierarchy generated by STROD (two phrases only differing in plural/single forms are shown only once)

Chapter 8

Conclusion and Future Work

This thesis investigates the problem of latent entity structure mining and propose methodologies to solve it. It first introduces and presents the background and preliminaries of the latent entity structure mining problems in unstructured and interconnected data, and then categorizes the two important structures, namely the topical and relational structures. Four subtasks are identified and both effective and efficient solutions are proposed and empirically validated on multiple real-world datasets, such as scientific publications, news articles, Web pages and social media.

The solution for the topical structure mining makes breakthrough in terms of quality and computational efficiency. The topics constituted by phrases and entities are much easier to interpret than traditional unigram-based topics. The first theoretically guaranteed method is developed to discover latent topic hierarchy recursively, and solve critical scalability challenges. The algorithm runs with several orders of magnitude faster speed than the state-of-the-art.

The solution for the relational structure mining is developped in a series of studies, and reached the state of the art in various domain tasks including academic publications, online forums and Web documents. Both unsupervised and supervised scenarios are studied. Heterogeneous signals including constraints and dependencies are categorized according to semantics, and formulated with simple and unified mathematical forms in the proposed solution.

In the following, we discuss several applications to showcase the impact of the thesis, and the future work.

8.1 Applications

This section, describes several representative applications related to mining latent entity structures.

8.1.1 Social influence and viral marketing

Word-of-mouth or viral marketing differentiates itself from other marketing strategies because it is based on trust among individuals' close social circle of families, friends, and co-workers. Research shows that people trust the information obtained from their close social circle far more than the information obtained from general advertisement channels such as TV, newspaper and online advertisements [63]. Thus many people believe that word-of-mouth marketing is the most effective marketing strategy [e.g. 62].

The increasing popularity of many online social network sites, such as Facebook, Myspace and Twitter, presents new opportunities for enabling large-scale and prevalent viral marketing online. Consider the following hypothetical scenario as a motivating example. A small company develops an online application and wants to market it through an online social network. It has a limited budget such that it can only select a small number of initial users in the network to use it (by giving them gifts or payments). The company wishes that these initial users would love the application and start influencing their friends on the social network to use it, and their friends would influence their friends' friends and so on, and thus through the word-of-mouth effect a large population in the social network would adopt the application. The problem is whom to select as the initial users so that they eventually influence the largest number of people in the network. One state of the art method for this problem is based on approximating the influence propagation with tree structures [86].

Influence maximization assumes information cascade as the information propagation model. Information cascade is essentially a tree structure of information propagation. The information receivers in the cascade form a hierarchical relationship. So mining hierarchical relationship can help identify the cascade, and provide accurate input to the influence maximization problem. For example, a social influence analysis method can use the cascade to determine the influence probabilities [80, 85].

Adopting the influence maximization idea in word networks, we can summarize the important topical words of a corpus. Further analysis of the influenced words by these topical words reveal the relations of the words. It provides an alternative solution to mining hierarchical relations and hierarchical communities [90].

8.1.2 Relevance targeting

Relevance targeting analyzes the data in online social networks and recommends users with ads and other information based on user interests. Relevance is the most important criterion in recommendation. We use ads for the illustration.

The foundation of pay-per-click advertising systems is a system for predicting the clickthrough rate of an ad for a given user or query. Typically, such predictions are based on a machine learning model that uses various hand-crafted features, and is trained on historical click data. Features employed in current social networking sites mainly inherit the keyword-targeting prototype that is successful in search engine advertising, and demography-targeted advertising that is prevalent in traditional brand advertising. However, the characteristics of social networking sites are not fully exploited by these methods. On one hand, the large amounts of data about user activity and social context are unique in social network services and offer the opportunity for better inference of personalized interests than traditional methods. For example, we can use user-generated content as contextual signals for content-based advertising. On the other hand, it is nontrivial to utilize these data. First, not all the contents have strong contextual relevance. For example, when users check friends' news or talk with friends, their intent is usually not related to business. Second, there is scalability issue if we simply concatenate all the text from linked objects to augment a user's profile. Third, if we use keyword or concept matching to measure the relevance independently of the targeting problem, simply counting the exact matching concepts or even related concepts in the taxonomy cannot capture the latent signal—for example, an ad about photographers is better targeted to users interested in weddings, rather than users interested in photography.

The hierarchical topics are useful to describe the interest of users and ads, so that they can be matched in a common feature space. In other words, mining the roles of users and ads in the hierarchical topics is the foundation of online targeting.

In a heterogeneous social network, users and ads are linked to different types of sources. The information in these sources does not always provide useful knowledge for targeting. It is important to mine the role of different sources in online targeting.

The underlying relations between the concepts in user space and ad space is also useful for relevance matching. For example, a user interest in wedding may have close relation with an ad concept photographer.

In Wang et al. [84], we develop a successful learning method that is applied to the social network data in Facebook. It relies on an existing topic hierarchy DMOZ/ODP hierarchy [1] as the feature space. It is expensive to maintain such a hierarchy by human. In the domain where an existing topic hierarchy is not available, obsolete or incomplete, the automatic learning of topic hierarchy will benefit such a targeting application.

8.2 Future work

Many real-world challenges remain unsolved. A long-term research goal is to build a well-structured "information map" for human-friendly navigation, search and reference. This has potential applications in a wide variety of disciplines, such as software engineering, bioinformatics and information security, in which data-driven analysis is rising as an important research approach. For example, can we discover the latent structures of gene, protein and diseases, so that the navigation of these entities on the "information map" could be as easy as location navigation on a geographic map?

A few research directions can be pursued:

1. User-guided structure discovery. The best information structure for different persons may vary according to their background knowledge and preference. For a specific domain like medicine, expert knowledge can be valuable to guide the mining process. Collaboration with specialists in HCI, AI, NLP, and other related fields, may enrich the experience of human-computer cooperation.

2. Evolutionary structures. The information in a certain domain may change at varying speeds, such as stock prices in finance data, or status updates in social media. Accurately analyzing such data requires swiftly adapting to the data stream and constantly incorporating up-to-date information.The infrastructure aspect is critical to building such a swift system, requring joint effort with experts in systems, networks and databases.

3. Structured information expansion, e.g. via Web and crowdsourcing. There are a number of data collections containing structured information, such as knowledgebases like Wikipedia, or

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

closed domain data repositories like Data.gov. It is possible to enrich the structured information provided by these collections with the less structured but high-volume open data from the Web or crowdsourcing. Future work would greatly benefit from cross-domain collaboration, since Web and crowdsourcing are already common subjects of study for researchers in different fields.

References

[1] Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *Proc. 2000 ACM conf. on Digital libraries (DL '00)*, 2000.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. VLDB*, volume 1215, pages 487–499, 1994.

[3] Amr Ahmed, Qirong Ho, Choon H Teo, Jacob Eisenstein, Eric P Xing, and Alex J Smola. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *AISTATS*, 2011.

[4] Amr Ahmed, Liangjie Hong, and Alexander Smola. Nested chinese restaurant franchise process: Applications to user tracking and document modeling. In *ICML*, 2013.

[5] Anima Anandkumar, Dean P Foster, Daniel Hsu, Sham Kakade, and Yi-Kai Liu. A spectral algorithm for latent dirichlet allocation. In *NIPS*, pages 926–934, 2012.

[6] Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559*, 2012.

[7] Animashree Anandkumar, Dean P. Foster, Daniel Hsu, Sham M. Kakade, and Yi-Kai Liu. A spectral algorithm for latent dirichlet allocation. *arXiv preprint arXiv:1204.6703*, 2012.

[8] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond svd. In *FOCS*, pages 1–10, 2012.

[9] Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. A practical algorithm for topic modeling with provable guarantees. In *ICML*, pages 280–288, 2013.

[10] Philip Bille. A survey on tree edit distance and related problems. *Theor. Comput. Sci.*, 337: 217–239, June 2005.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[12] D. M. Blei and J. D. Lafferty. Visualizing topics with multi-word expressions. *arXiv preprint arXiv:0907.1013*, 2009.

[13] David M Blei and Michael I Jordan. Modeling annotated data. In *SIGIR*, pages 127–134, 2003.

[14] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[15] K. Bollacker, R. Cook, and P. Tufts. Freebase: A shared database of structured general human knowledge. In *AAAI '08*, 2008.

[16] Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Math. Program.*, 63:129–156, 1994.

[17] Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. Reading tea leaves: How humans interpret topic models. In *NIPS*, 2009.

[18] Xu Chen, Mingyuan Zhou, and Lawrence Carin. The contextual focused topic model. In *KDD*, 2012.

[19] Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Wen-Pin Lin, Javier Artiles, Matthew Snover, Marissa Passantino, and Heng Ji. Cuny-blender tac-kbp2010 entity linking and slot filling system description. In *Proc. 2010 NIST Text Analytics Conference (TAC '10)*, 2010.

[20] Kenneth Church, William Gale, Patrick Hanks, and Donald Kindle. chapter 6. using statistics in lexical analysis. *Using statistics in lexical analysis*, page 115, 1991.

[21] Bonaventura Coppola, Alessandro Moschitti, and Daniele Pighin. Generalized framework for syntax-based relation mining. In *ICDM*, pages 153–162, 2008.

[22] Aron Culotta, Andrew McCallum, and Jonathan Betz. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proc. HLT Conf. the North American Chapter of the Association of Computational Linguistics (HLT-NAACL '06)*, 2006.

[23] Marina Danilevsky, Chi Wang, Nihit Desai, and Jiawei Han. Entity role discovery in hierarchical topical communities. In *KDD workshop MDS*, 2013.

[24] Marina Danilevsky, Chi Wang, Nihit Desai, Jingyi Guo, and Jiawei Han. Automatic construction and ranking of topical keyphrases on collections of short documents. In *SDM*, 2014.

[25] Hongbo Deng, Jiawei Han, Bo Zhao, Yintao Yu, and Cindy Xide Lin. Probabilistic topic models with biased propagation on heterogeneous information networks. In *KDD*, 2011.

[26] Luigi Di Caro, K. Selçuk Candan, and Maria Luisa Sapino. Using tagflake for condensing navigable tag hierarchies from tag clouds. In *KDD '08*, 2008.

[27] Christopher P. Diehl, Galileo Namata, and Lise Getoor. Relationship identification for social network discovery. In *AAAI*, 2007.

[28] "Ahmed El-Kishky", Yanglei Song, Chi Wang, and Clare R. Vossand Jiawei Han. Scalable topical phrase mining from text corpora. *arXiv preprint arXiv:1406.6312*, 2014.

[29] James Foulds, Levi Boyles, Christopher DuBois, Padhraic Smyth, and Max Welling. Stochastic collapsed variational bayesian inference for latent dirichlet allocation. In *KDD*, 2013.

[30] Brendan J. Frey. *Graphical models for machine learning and digital communication.* MIT Press, Cambridge, MA, USA, 1998. ISBN 0-262-06202-X.

[31] Benjamin CM Fung, Ke Wang, and Martin Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of SIAM international conference on data mining*, pages 59–70, 2003.

[32] T Griffiths, M Jordan, J Tenenbaum, and David M Blei. Hierarchical topic models and the nested chinese restaurant process. *NIPS*, 2004.

[33] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[34] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *ACL '05*, 2005.

[35] Michael AK Halliday et al. Lexis as a linguistic level. *In memory of JR Firth*, pages 148–162, 1966.

[36] J.M. Hammersley and P. Clifford. Markov field on finite graphs and lattices, 1971. Unpublished.

[37] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, volume 29, pages 1–12. ACM, 2000.

[38] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques.* 3rd ed., Morgan Kaufmann, 2011.

[39] M Hoffman, D Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[40] Matt Hoffman, David M Blei, and David M Mimno. Sparse stochastic inference for latent dirichlet allocation. In *ICML*, 2012.

[41] David Cohn Thomas Hofmann. The missing link-a probabilistic model of document content and hypertext connectivity. In *NIPS*, pages 430–436, 2001.

[42] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, January 2001.

[43] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002. ISSN 1046-8188.

[44] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *ACL '11*, 2011.

[45] Charles Kemp and Joshua B B. Tenenbaum. The discovery of structural form. *Proceedings of National Academy of Sciences of the United States of America*, July 2008.

[46] Hyun Duk Kim, Dae Hoon Park, Yue Lu, and ChengXiang Zhai. Enriching text representation with frequent pattern mining for probabilistic topic modeling. *Proceedings of the American Society for Information Science and Technology*, 49(1):1–10, 2012.

[47] Hyungsul Kim, Yizhou Sun, Julia Hockenmaier, and Jiawei Han. Etm: Entity topic models for mining documents associated with entities. In *ICDM*, 2012.

[48] Joon Hee Kim, Dongwoo Kim, Suin Kim, and Alice Oh. Modeling topic hierarchies with the recursive chinese restaurant process. In *CIKM*, 2012.

[49] Frank R. Kschischang, Senior Member, Brendan J. Frey, and Hans andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 2001.

[50] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD '09*, 2009.

[51] Q. Li, S. Anzaroot, W. Lin, X. Li, and H. Ji. Joint Inference for Cross-document Information Extraction. In *CIKM '11*, 2011.

[52] Qi Li, Heng Ji, and Liang Huang. Joint event extraction via structured prediction with global features. In *ACL*, 2013.

[53] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML*, 2006.

[54] Robert V. Lindsey, William P. Headden, III, and Michael J. Stipicevic. A phrase-discovering topic model using hierarchical pitman-yor processes. In *EMNLP-CoNLL*, 2012.

[55] Xueqing Liu, Yangqiu Song, Shixia Liu, and Haixun Wang. Automatic taxonomy construction from keywords. In *KDD*, 2012.

[56] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proc. EMNLP*, pages 257–266, 2009.

[57] Arun S. Maiya and Tanya Y. Berger-Wolf. Inferring the maximum likelihood hierarchy in social networks. In *Proc. 2009 Int. Conf. on Computational Science and Engineering*, 2009.

[58] Andrew McCallum, Andrés Corrada-Emmanuel, and Xuerui Wang. Topic and role discovery in social networks. In *IJCAI*, 2005.

[59] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[60] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, page 275, 2004.

[61] David Mimno, Wei Li, and Andrew McCallum. Mixtures of hierarchical topics with pachinko allocation. In *ICML*, 2007.

[62] Ivan R. Misner. *The World's best known marketing secret: Building your business with word-of-mouth marketing.* Bard Press, 2nd edition, 1999.

[63] Jim Nail. The consumer advertising backlash, May 2004. Forrester Research and Intelliseek Market Research Report.

[64] David Newman, Chaitanya Chemudugunta, and Padhraic Smyth. Statistical entity-topic models. In *KDD*, pages 680–686, 2006.

[65] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009.

[66] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence. In *NAACL-HLT*, 2010.

[67] Ted Pedersen. Fishing for exactness. *arXiv preprint cmp-lg/9608010*, 1996.

[68] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *KDD*, 2008.

[69] Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[70] Jay Pujara and Peter Skomoroch. Large-scale hierarchical topic models. In *NIPS Workshop on Big Learning*, 2012.

[71] Dan Roth and Wen-tau Yih. Probabilistic reasoning for entity & relation recognition. In *Proc. 2002 ICCL Conf. on Computational Linguistics (COLING '02)*, 2002.

[72] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[73] J. Seo, W.B. Croft, and D.A. Smith. Online community search using thread structure. In *CIKM '09*, 2009.

[74] Alexander Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, 2010.

[75] Padhraic Smyth. Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and Computing*, 10(1):63–72, 2000.

[76] David Sontag and Dan Roy. Complexity of inference in latent dirichlet allocation. In *NIPS*, pages 1008–1016, 2011.

[77] Yizhou Sun, Jiawei Han, Jing Gao, and Yintao Yu. itopicmodel: Information network-integrated topic modeling. In *ICDM*, 2009.

[78] Yizhou Sun, Yintao Yu, and Jiawei Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD*, 2009.

[79] Jian Tang, Ming Zhang, and Qiaozhu Mei. One theme in all views: modeling consensus topics in multiple contexts. In *KDD*, 2013.

[80] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *KDD*, 2009.

[81] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment - Volume 18*, MWE '03, 2003.

[82] Hanna M. Wallach. Topic modeling: beyond bag-of-words. In *ICML*, 2006.

[83] Chi Wang, Jiawei Han, Yuntao Jia, Jie Tang, Duo Zhang, Yintao Yu, and Jingyi Guo. Mining advisor-advisee relationships from research publication networks. In *KDD*, 2010.

[84] Chi Wang, Rajat Raina, David Fong, Ding Zhou, Jiawei Han, and Greg Badros. Learning relevance from heterogeneous social network and its application in online targeting. In *SIGIR*, 2011.

[85] Chi Wang, Jie Tang, Jimeng Sun, and Jiawei Han. Dynamic social influence analysis through time-dependent factor graphs. In *ASONAM*, 2011.

[86] Chi Wang, Wei Chen, and Yajun Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3): 545–576, 2012.

[87] Chi Wang, Jiawei Han, Qi Li, Xiang Li, Wen-Pin Lin, and Heng Ji. Learning hierarchical relationships among partially ordered objects with heterogeneous attributes and links. In *SDM*, 2012.

[88] Chi Wang, Marina Danilevsky, Nihit Desai, Yinan Zhang, Phuong Nguyen, Thrivikrama Taula, and Jiawei Han. A phrase mining framework for recursive construction of a topical hierarchy. In *KDD*, 2013.

[89] Chi Wang, Marina Danilevsky, Jialu Liu, Nihit Desai, Heng Ji, and Jiawei Han. Constructing topical hierarchies in heterogeneous information networks. In *ICDM*, 2013.

[90] Chi Wang, Xiao Yu, Yanen Li, Chengxiang Zhai, and Jiawei Han. Content coverage maximization on word networks for hierarchical topic summarization. In *CIKM*, 2013.

[91] Chi Wang, Xueqing Liu, Yanglei Song, and Jiawei Han. Scalable moment-based inference for latent dirichlet allocation. In *Machine Learning and Knowledge Discovery in Databases*, pages 290–305. Springer, 2014.

[92] Hongning Wang, Chi Wang, ChengXiang Zhai, and Jiawei Han. Learning online discussion structures by conditional random fields. In *SIGIR*, 2011.

[93] Xuerui Wang, Andrew McCallum, and Xing Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *ICDM*, 2007.

[94] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47: 723–735, 2001.

[95] Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology learning from text: A look back and into the future. *ACM Computing Surveys (CSUR)*, 44(4):20, 2012.

[96] Tianyi Wu, Yuguo Chen, and Jiawei Han. Re-examination of interestingness measures in pattern mining: A unified framework. *Data Mining and Knowledge Discovery*, 2010.

[97] Zi Yang, Jie Tang, Bo Wang, Jingyi Guo, Juanzi Li, and Songcan Chen. Expert2bole: From expert finding to bole search (demo paper). In *KDD'09*, 2009.

[98] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD*, 2009.

[99] Xiaoxin Yin and Sarthak Shah. Building taxonomy of web search intents for name entity queries. In *WWW '10*, 2010.

[100] Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad L Alkhouja. Mr. lda: A flexible large scale topic modeling package using variational inference in mapreduce. In *WWW*, 2012.

[101] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. Topical keyphrase extraction from twitter. In *ACL-HLT*, 2011.