ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Joint Inference of Concepts and Networks of Documents

Master Thesis

Andreas Tschofen

March 31, 2013

Advisors: Prof. Dr. Andreas Krause, Prof. Dr. Donald Kossmann, Adish Singla

Department of Computer Science, ETH Zürich

Acknowledgements

I would like to thank Professor Andreas Krause and Professor Donald Kossmann for the guidance and inspiration throughout the project. Their big effort allowed me to learn a lot. Furthermore, I would like to thank Adish Singla for his commitment, the helpful discussions and good advice. I am deeply grateful to my parents and my brothers for their unconditional love and support.

Abstract

Due to the growing number of publications, scientists face the problem of information overload. Getting an overview of a new research area, finding relevant related work, and grasping connections between different publications are essential but difficult tasks. Recent efforts have been directed towards the automatic discovery of structure in document collections that can help scientists orientate in a research domain. In this thesis, we investigate the problem of finding semantic relationships between documents based on their textual content. In particular, we take an epidemiological view of information diffusion. We assume that documents are connected in a graph of influence, and ideas travel over this graph from document to document. Knowledge of the graph could give valuable information about the relationship between documents and help a user navigate a document collection. Thus, we address the task of inferring a document influence graph.

We track ideas by memes - keywords, technical terms or short phrases of technical terminology - that appear in the document text. Based on a model of the diffusion of memes, we address the task of recovering the document graph from the observations of many memes in the documents. In a further step, we develop a more realistic model in which the contagions are not simple memes but concepts - sets of related memes - that can mutate in the diffusion process. We present the DocNETINF algorithm to jointly infer concepts and the document network. By experiments on simulated data, we show that DocNETINF can recover ground truth graph and concepts better than baseline methods. We apply our algorithm to scientific papers from the CiteSeerX dataset. To this end, we describe a simple method to extract memes from the paper text. Finally, we present a demo application that allows to use the inferred graph and concepts for exploratory purposes. An anecdotal analysis gives the reader a flavor of the results.

Contents

Chapter 1

Introduction

This section gives an introduction and a high-level overview of the contents of this thesis. We first discuss the problem of *information overload* in scientific document collections which motivates our goal of inferring semantic relationships between documents. We introduce the concept of *information diffusion* which underlies our approach. Then, our main contributions, a model and an algorithm to jointly infer concepts and a network of scientific documents, are summarized. Finally, we give an outline of the thesis.

1.1 Information Overload in Scientific Document Collections

As the number of conferences and published papers grows, scientists are confronted with an *information overload* problem: orientating in a scientific domain has become a difficult task. Traditional techniques like keyword-based search which is offered by platforms like Google Scholar[1] or CiteSeerX[2] have evolved to great quality and are the predominant tools for information retrieval tasks in the scientific domain to date. However, several important tasks of exploratory nature, e.g., getting an overview of a scientific domain, understanding developments and connections, and finding related work, cannot be effectively supported by keyword search alone. To help us understand the big picture, we need tools to extract structure from large document collections and make them navigable.

Recently, a line of research has been investigating tasks related to discovering coarse-grained structure and developments in scientific and non-scientific document collections ([27], [26]). In particular, the authors propose a method to generate *information maps* that can help a user to discover seminal docu-

[1] scholar.google.com
[2] http://citeseerx.ist.psu.edu/

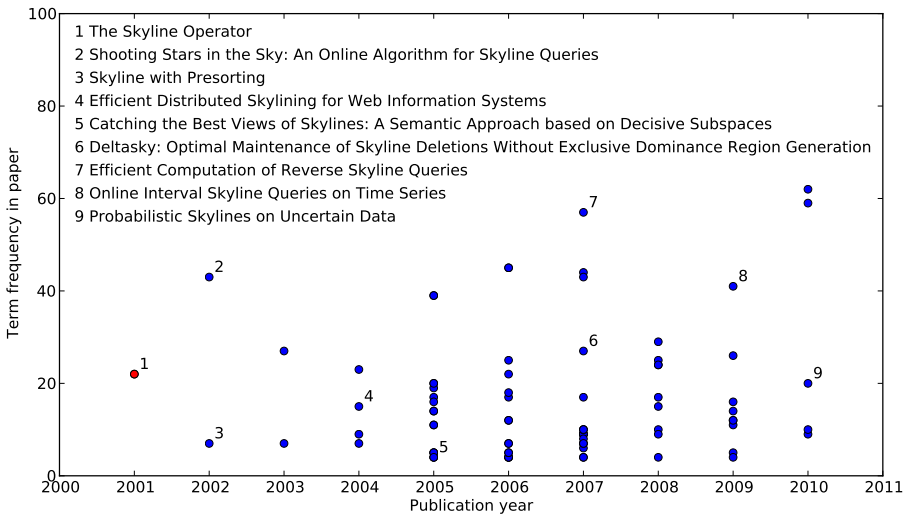


Figure 1.1: Papers mentioning the concept *skyline query*. The red point represents the paper *The Skyline Operator* that introduced the concept, the blue points are later papers that elaborated on, extended, or applied it. We assume that the idea of a skyline query has diffused over a hidden network of influence between the papers.

ments and coherent lines of research. In our work, we address the extraction of a more fine-grained structure from scientific document collections, namely a network of documents, where the links indicate semantic relationships, and are annotated with the concepts constituting this relationship. In the scientific domain, we often actually have a citation graph, where documents are connected by a link if one cites the other, that represents influence between documents. Several tools have been developed to allow scientists navigate this graph and leverage it to discover relevant papers ([10], [1]). While citation links are clearly essential indicators for the relationship between documents and can be utilized to enable the exploration of paper datasets, their usage in exploration tools suffers from certain limitations: First of all, citations are not always available, since their automatic extraction from paper texts is a difficult task. Furthermore, a standard conference paper usually cites dozens of other papers, not all of which have had an essential influence on the presented work. Finally, citations per se are not annotated, i.e., to find the reason for a paper citing another, the user usually has to consult the paper text. For this thesis, we thus do not use citation links, but address the task of inferring a *network of diffusion and influence* of scientific papers based purely on the textual content of documents.

1.2 Diffusion of Information

To solve the task at hand, we take a viewpoint on the creation of scientific literature inspired by mathematical epidemiology [3]. In his famous book "Diffusion of Innovations" [23], E. M. Rogers established the theory that innovations proliferate by diffusion processes through the members of a society similar to viruses. This idea has been widely applied under the general term *diffusion of information* including recommendation influences [18] and the spread of news in blogspace ([2], [12]). Likewise, we consider scientific pieces of information, ideas, concepts, etc. as contagions, infecting certain authors when they write a document. The semantic relationships we would like to discover are links of influence over which these contagions diffuse from one document to another.

The basic unit of contagions we track are *technical terms*, words or combinations of words describing a concept or an aspect of a concept in a certain domain. We assume that these technical terms are *memes*, i.e., basic units of text that are the trace of an idea, of the elaboration on or the application of the corresponding domain concept. Consider as an example the technical term *skyline query*. It was introduced in 2001 in the paper *The Skyline Operator* by S. Borzsony et al. and describes a database operator that should return a tuple that dominates all other considered tuples w.r.t. a certain set of attributes (called the *Pareto frontier* in economics). From this point on, many authors picked up the concepts and worked on efficient implementations, extensions, variations, analyses, etc. Figure 1.1 shows a plot of papers mentioning *skyline query*. Points represent papers, the x-axis corresponds to time, and on the y-axis we show the occurrence frequency of the technical term in the corresponding paper. Some of the paper titles are given for illustration. One can see from the plot that shortly after the concept was introduced in 2001, there were only few papers on the topic, but then the concept diffused through the scientific community and many papers presented work on the topic. Towards 2010, the trend seems to flatten out a little bit.

The basic assumption of our work is that papers are connected in a hidden influence graph, and that memes like *skyline query* traveled over this graph from one paper to another. A potentially more natural way of putting this is the following: We assume that when a paper is written, the authors first choose some influencing papers (this causes influence edges from these papers to the new paper to be added to the graph), and then adopt concepts from these influencing papers (these concepts travelled over the corresponding edge). Together with original concepts added by the authors, the set of all these concepts defines the new document. The goal of our work is to infer this hidden graph. In a first simple model, we directly consider memes to be concepts. Later on, we investigate a more natural model where concepts consist of several terms (e.g., a basic skyline concept: *skyline*, *skyline query*,

and *skyline operator*). In this model, we address the task of jointly inferring concepts and the network of diffusion.

Although a model of the generation of scientific papers as described above is clearly a strong simplification of the real process, we believe that it is basically valid and captures some essential aspects. In particular, we believe that a diffusion graph with concept-annotated edges can capture and reveal interesting semantic relationships between documents.

1.3 Thesis Contribution

We develop a simple model of how memes travel through a network of documents. We show that it can be interpreted as a version of the *independent cascade model* [14] on a graph of documents. To solve the network inference problem, we take a maximum likelihood approach to derive an objective function. This objective function is shown to be a special case of the objective function of the NETINF algorithm [12], an algorithm designed to recover a network of nodes based on the observation of cascades. Consequently, we can utilize NETINF to infer a diffusion network of documents.

Our main contribution is the following: We develop a more complex diffusion model for concepts which are sets of memes. When a document is infected with a concept, a subset of its memes makes up the *realization* of the concept in a document. In the diffusion process, the realizations change according to a *Markov process*, simulating the notion of *mutation* from one occurrence of a concept to another. The concepts are not known in advance. Thus, we develop DocNETINF, an algorithm to infer concepts and the corresponding diffusion network based on a joint objective function. DocNETINF alternates between inferring a network for fixed concepts, and clustering memes to concepts for a fixed network, to find a local optimum of the objective function. We simulated data according to our model and show that the algorithm can recover both concepts and network with high accuracy.

We describe a simple method to extract technical terminology, the memes, from the plain text of a document corpus. We consider n-grams of length two and three and filter out n-grams that start or end with a stopword. As a measure of *termhood* for the remaining candidate n-grams, we use the maximal frequency of the candidate in a document. Furthermore, we employ a list of words with their frequency in standard English texts to detect unigrams that are extraordinarily frequent and thus also keywords for the document at hand.

We apply DocNETINF to scientific papers from the CiteSeerX dataset and show that the results are meaningful by an anecdotal analysis. Finally, we present a demo application that allows a user to explore the inferred connections between documents and follow the diffusion of concepts (cf. Figure

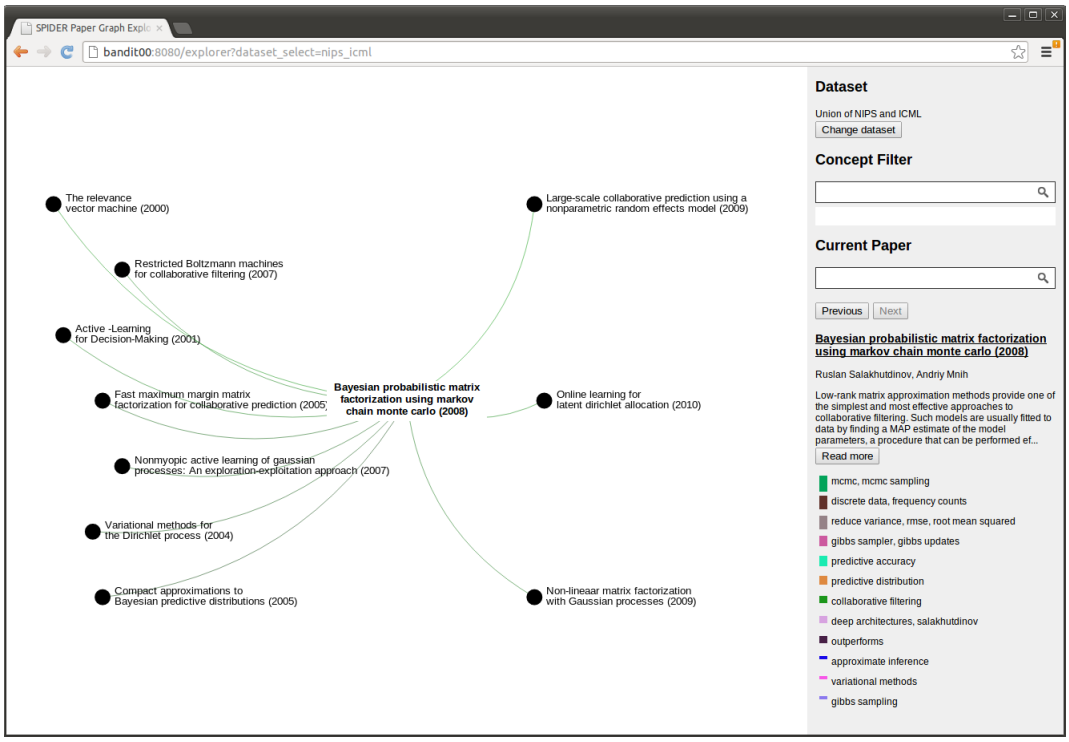


Figure 1.2: Screenshot of the demo application, a web application that allows for the navigation of the document network and associated concepts.

1.2 for a screenshot). Two professors at our institution experimented with the demo application and their feedback was generally positive.

1.4 Thesis Outline

The rest of this thesis is structured as follows: In Section 2, we discuss related work in terms of the application we have in mind as well as methods we apply or build on. The basic document network inference problem is addressed in Section 3. In Section 4, we develop a more expressive model of concept diffusion and formally state the problem of jointly inferring concepts and a network of documents. DocNETINF, the algorithm we propose to solve the problem, is described in Section 5. In Section 6, we present the results of our experiments on simulated data. Finally, the application to real data including the extraction of technical terms from documents and the demo application is described in Section 7.

Chapter 2

Related Work

In this section, we describe related work in terms of application and our technical approach. In particular, we discuss existing approaches for discovering structure in document collections. Then, we briefly review automatic term recognition, topic modeling, and general link prediction and network inference.

2.1 Finding Structure in Document Collections

Motivated by the growth of both scientific and non-scientific document collections and the arising information overload, recent research efforts have been aiming at the automatic detection of structure that can help users discover general themes, understand connections between documents, and find relevant literature.

The work in [28] by B. Shaparenko et al. addresses the task of uncovering influence between non-hyperlinked documents based on their textual content. The authors establish a model of how the word distribution of a document is generated as a mixture of the word distribution of influencing documents and new parts. To decide if a paper had a significant influence on another, they employ a statistical likelihood ratio test. Their experiments show that they outperform simple bag-of-words similarity-based baselines in the tasks of inferring citations and publication impact.

K. El-Arini et al. [9] addressed the problem of searching for relevant scientific literature by a small set of example papers. To this end, they define a notion of influence between scientific papers that is based on assigning influence weights w.r.t words to edges in the citation graph. The probability of influence of a paper u on a paper v is then formalized as the probability for the existence of a path consisting of only active edges between u and v, where an edge is active with a probability proportional to the correspond-

ing influence weight. Computing this probability exactly is intractable, but the authors show two possibilities to approximate it: a sampling-based approach and a heuristic assuming independence of influence paths. In the final objective function, the defined notion of influence is used as an indicator for relevance. As it obeys the submodularity property, a near-optimal argument of the maximum can be found efficiently. The approach was evaluated in a user study in which participants had to rate the usefulness and diversity of result sets of the proposed method and baseline methods. The outcomes suggest that the *Beyond Keyword Search* yields better results.

D. Shahaf et al. [25] proposed a method that solves the following task: Given two news articles, find a semantically coherent chronological chain of news articles that connect the two endpoints, i.e., represents the development of events that led from the older the the newer story. The used data is the textual content of articles only. They formalize the coherence of a chain, requiring a "non-jittery" activation pattern of words and no weak links for a coherent chain. They compute the influence of a document u on another document v with respect to a word w with a random walk simulation on a bipartite graph of words and documents. Consequently, influence between documents does not have to show by the use of the exact same vocabulary, but can also be exerted through related words. The problem of finding a good chain is formulated as a linear programming problem that can be solved efficiently. With a user study, the authors demonstrate that their approach outperforms Google News Timeline[1] and baselines in terms of perceived coherence of the chains and improvement in familiarity with the story.

Recently, the authors built on this work and developed algorithms to generate so-called "metro maps" of news stories [27] and scientific domains [26]. We focus on [26] since the application is more related to our work. A metro map of a scientific domain should contain influential papers connected in coherent lines of research. The authors reuse the notion of coherence, and additionally incorporate the citation graph as an indicator for influence between papers. Furthermore, they add terms for *coverage* (a good map should cover the important topics of a domain) and *connectivity* (papers on the map should be highly connected for the map to be informative) to their objective function. To optimize it, they proceed in a number of steps: Coherent chains are represented as a graph. Since the notion of coverage is submodular, they can apply a *submodular orienteering algorithm* to find a set of chains with high coverage. Finally, they increase connectivity by local search. In a user study, participants were asked to update an outdated survey with new influential papers. The results suggest that the task can be solved better with

[1] news.google.com

the Google Scholar[2] search engine and a metro map for the corresponding science domain than with Google Scholar alone.

2.2 Automatic Term Recognition

Automatic term recognition is the process of extracting technical terms, which we use as the basic unit representing ideas or concepts (so-called *memes*) for this work, from a document collection. Several methods for term recognition have been proposed in the literature. Most of them basically use linguistic filters to retrieve term candidates and then score potential candidate terms based on a measure of *termhood*, *unithoood* or a combindation of these. Termhood measures the likelihood that a given candidate represents a domain concept, while unithood measures, for a term candidate consisting of multiple words, the attachment strength of the words [15].

The *C-value* method [11] is a termhood-based method for term extraction, some ideas of which we adopt for our approach. The method consists of the following steps: First, a linguistic filter is applied to only extract certain sequences of word types (e.g., noun phrases). Then, each candidate's termhood is computed based on frequency counts. In the simplest case, the termhood score could be just the plain frequency of the candidate in the corpus. For the actual C-value, however, the average occurrence frequency of the candidate as part of a longer candidate is subtracted to avoid subterms being extracted as proper terms. Finally, the value is scaled with the logarithm of the term length to favor longer over shorter terms. In the same paper, the authors propose an extension to the C-value method, the *C-value/NC-value method*, which also incorporates automatically extracted context words that suggest termhood (the authors mention e.g., *present*, *shows*, *composed* for a corpus of medical documents) into the term recognition. They show that their methods yield slightly better precision than the simple term frequency termhood score on a corpus of medical documents.

2.3 Topic Modeling

The topic modeling field deals with the extraction of *topics* from natural language texts. A topic is a distribution over words that belong to a common theme, which corresponds to the intuitive idea. Although our notion of a *concept* is more fine-grained and we represent concepts as sets instead of distributions, topics and concepts are still similar in the sense that they both should group semantically related terms.

Topic models are latent variable models. Documents are modeled as distributions over topics (the latent variables), which are in turn distributions over

[2] scholar.google.com

words. Fitting a topic model to a document collections corresponds to the estimation of these distributions. This is usually done by either *variational methods* or *Markov chain sampling* [4].

In [7], D. Blei et al. introduce *Latent Dirichlet Allocation* (LDA), the first topic modeling approach based on a complete generative model of documents. LDA is widely used and has built the basis for many new topic models [4].

J. Chang et al. extended LDA to the *Relational Topic Model* (RTM) [8]. It is a generative model for documents and the links between them. In particular, a document's content is assumed to be generated from the corresponding mixture of topic distributions as in LDA, and the link between a pair of documents u and v is modeled as a logistic regression for a binary random variable whose value depends on the topic assignments of the words in u and v. The authors show that their method outperforms baselines with no joint model for topics and links in the tasks of link prediction (based on document content) and word prediction (based on document links).

C. Lin et al. [20] investigated topic diffusion and evolution in social networks, combining topic modeling with epidemiological models of information diffusion as we use them in this thesis. Based on a social network of authors and their documents, they infer how a given topic diffused through the community (the diffusion graph) and how it evolved in the process (time-variant versions of the topic). Their *TIDE* model is a probabilistic model that can be conceptually divided into the topic model and the diffusion model. In the topic model, a document d's content is modeled as a mixture of a background model (content not related to the tracked topic) and components for all documents that could have influenced d. For the diffusion model, each document is modeled as a diffusion vector, and the diffusion graph is regularized by the social network of authors. The parameters are estimated with an *Expectation Maximization* algorithm. By performing experiments on data simulated according to their model, the authors show that their approach outperforms baseline methods in both diffusion graph and time-variant topic version inference. Furthermore, they apply their method on real-world datasets of scientific papers and tweets. The quality of the results is rated higher by an export than the results of their baselines.

2.4 Link Prediction and Social Network Inference

Hidden link prediction in different kinds of networks is a widely studied area of research. B. Taskar et al. [29] applied the relational Markov network framework to infer links between university webpages and students in a small friendship network of students. D. Liben-Nowell et al. [19] formalized the link prediction task in social networks and showed that future ties can be predicted based on topological information of the current social network

graph only. Variants of the task like predicting relationship strength, and positive (e.g., friendship) or negative (e.g. distrust) connotation of links, have been investigated in [31] and [17], respectively.

While in the beforementioned papers, link prediction is formulated as a supervised learning problem, more recent work has addressed the unsupervised inference of networks ([12], [21]). In these approaches, links in an unobserved network are inferred based on the observation of many contagions that spread through the nodes of the network according to a certain diffusion process. One instance of such a diffusion is called a *cascade*. A helpful epidemiological analogy is a social network of friends. The contagions are viruses that infect certain nodes, and infected nodes can in turn infect the nodes they interact with (their friends). Intuitively, the number of times two nodes are infected with the same virus, and the time between their infections, respectively, contain information about their proximity in the network. So the network inference task can be stated as follows: Given observed cascades (i.e., data about who was infected with which virus at what time), infer the social network. In general, the model applies to many similar real-life processes like the spread of news through the blogosphere [12], or the spread of ideas and concepts through a scientific community.

The most common diffusion models are the *Linear Threshold Model* and the *Independent Cascade Model* and variants thereof. They were defined by Kempe et al. [14] in the context of a related problem, the influence maximization problem. In both these models, a node can either be active (infected) or inactive (not infected) with respect to a contagion. At each discrete timestamp, currently inactive nodes can become active, and active nodes always stay active. The Linear Threshold Model and the Independent Cascade Model differ in the conditions for a node to become active. In the Linear Threshold Model, each edge (u, v) is associated with an influence weight $b_{u,v}$. A node becomes active when the sum of the influence weights of all its active neighbors exceeds a certain threshold. In the Independent Cascade Model, a node that becomes active gets a change to influence each of its neighbors. With probability $p_{u,v}$, a node u is successful with influencing v. A node becomes active as soon as the first influence attempt on it is successful.

Two representatives of the described approach to network inference are the NETINF [12] and ConNIe [21] algorithms. In [12], M. Gomez et al. introduce a variant of the independent cascade model that incorporates time. In particular, the diffusion process does not progress in discrete timestamps. Instead, the time between infection and outbreak is assumed to follow an exponential or power-law distribution. Based on this model, they formulate the network inference problem as a maximum likelihood estimation problem, i.e., finding a network structure such that the observed cascades have maximum likelihood. They state an approximate variant of this objective function that

is a *submodular* function in the set of selected edges. Consequently, it can be optimized by a greedy algorithm with near-optimal performance. In this thesis, we apply and build on the NETINF algorithm to infer networks of documents.

Similarly, A. Meyers et al. [21] also state the problem as a maximum likelihood estimation problem. However, instead of choosing the optimal edge set for the network, they find infection probabilities for each pair of nodes (a vanishing infection probability corresponds to no edge). They give a convex objective function that is equivalent to the cascade likelihood and allows for an efficient solution. Both approaches are evaluated on randomly generated graphs and data arising from simulating cascades. Perhaps surprisingly, both methods can recover the unobserved network almost perfectly under a broad range of conditions.

Chapter 3

Document Network Inference

In this section, we describe our model of how scientific documents are generated according to a diffusion process of ideas. Intuitively, we assume that when authors decide to write a document, they choose a certain number of influencing documents. From each of these documents, they adopt some ideas, and add their original ones. We assume that ideas manifest themselves in documents as *memes*.

Definition 1 *A meme is a term, a multi-word term, or a short phrase that describes a certain technical concept or idea. It corresponds roughly to the notion of a technical term.*

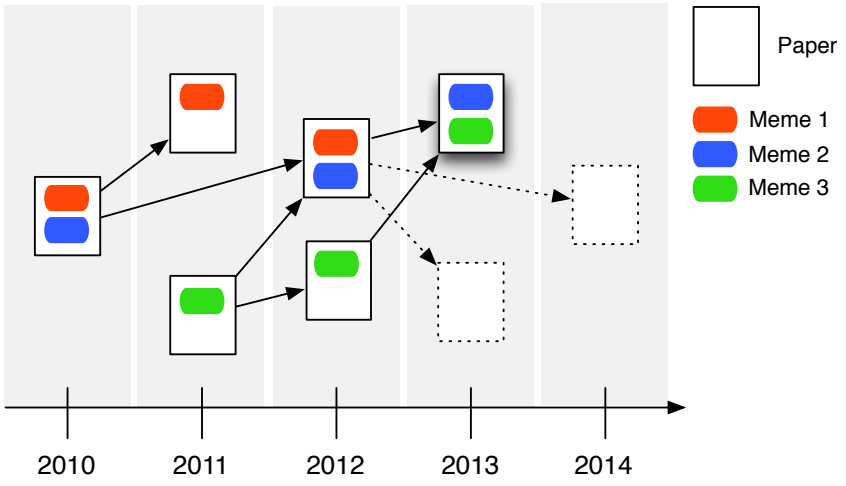


Figure 3.1: Document Network. The memes are introduced by certain documents, diffuse over the graph, and become part of the content of later documents.

Memes related to this thesis are for example *document network*, *diffusion of information*, *infer network*, or *meme* itself. Based on this definition, we model a document as a set of the adopted memes. Furthermore, the choices of influencing documents constitute a graph of influence between documents, where edges point from influencing to influenced documents. Figure 3.1 illustrates this model. The dotted documents and arrows indicate parts of the graph that do not yet exist at the current point in time. The colored rectangles represent memes that are introduced or adopted from influencing documents. One can think of them as traveling over the influence graph from document to document.

We assume the influence graph to be hidden, and address the problem of inferring it based on the observation of the document memes. To this end, we formally describe our model in Section 3.1, and give the corresponding problem statement in Section 3.2. We show that the problem reduces to a more general network inference problem which can be solved with the existing NETINF algorithm.

3.1 Independent Cascade Model for Document Networks

We consider a directed document graph G^* with vertex set $V = \{d | d \text{ a document}\}$ and edge set E. Each document is associated with a timestamp (its creation time) t_d. For each edge $(d, d') \in G^*$ it must hold that $t_d < t_{d'}$, i.e. edges can only point from older to newer documents, since documents can only be influenced by documents that have already existed at their time of creation. Note that this requirement implies that G^* is a *directed acyclic graph*. Any cycle would have to contain at least one edge (d, d') with $t_d > t_{d'}$. We define the set of neighbors of a document d in the graph as $N(d) = \{d' \in V | (d, d') \in E\}$. $N(d)$ contains all documents that are potentially influenced by d.

We think of memes travelling over this graph of influence between documents in the following way: If a document d has adopted a meme m, it gets one chance to *infect* each of its currently uninfected neighbors from $N(d)$ with m. This infection is successful with a constant probability β, a parameter of the model. If this is the case for a neighboring document d' of d, at the creation time $t_{d'}$ of d' (which is after the creation of d since $t_d < t_{d'}$), d' will adopt meme m and will itself get the chance to infect its neighbors with m. Note that after the first successful infection of a document d with a meme m, it cannot be infected with m again. One successful infection with m determines its adoption.

We assume that all memes from a meme vocabulary M diffuse over the document graph as described above. After the process has finished, each

document d has been infected with a set $M_d \subseteq M$. This set M_d defines the content of d in the sense that it contains all memes adopted by d. The set of documents that have adopted meme m constitute an information cascade c, and the set of all cascades is C.

Note that this model of a document graph that exists in advance and concepts that diffuse over it is equivalent to the model where a document d and all edges (d', d) are added to G^* at the document creation time t_d. Instead of thinking of the nodes d' with $d \in N(d')$ as infecting d, one can say that the authors of d decide about the adoption of each meme in d' with probability β, such that the meme set M_d is defined at once at time t_d. While this might be a more intuitive interpretation, we focus on the first version for two reasons: First, the notation is more convenient since we assume a fixed graph instead of a time series of graphs. And second, the initial formulation is similar to cascade models described in the literature, especially the original *independent cascade model* [14] and the version used in the model for the NETINF algorithm for network inference [12].

The independent cascade model for the NETINF algorithm assumes a general graph. Nodes have no timestamps. Instead, each infection event of a node with a contagion is associated with a timestamp. When a node is infected with a contagion, it can infect its neighbors similarly as described above. The difference between the models is that the time of *adoption* (the point in time when an infected node gets a chance to infect its neighbors) is assumed to follow a probability distribution (the exponential and the power-law distribution are investigated), while in our case, the time of adoption is fixed to be the timestamp of the document.

3.2 Problem Statement and Reduction to NETINF

The task we would like to solve is the following: Given a set of documents V, timestamps t_d and meme sets M_d for all documents $d \in V$, infer the graph G^*. In this section, we derive a formula for the likelihood of a graph G which can be maximized to find G^*. We show that this likelihood reduces to the likelihood formulation derived for NETINF, meaning that we can apply the NETINF algorithm to solve the problem.

Let us first focus on the likelihood of an individual cascade c for a meme m. Since the cascades are independent, the joint likelihood of all cascades is just the product of the individual likelihoods. Note that c is associated with an unobserved diffusion graph $T = (V_T, E_T)$ with V_T the set of documents infected with m, and $E_T \subseteq E$ the edges over which m spread. Since a document can only be infected with a meme once, T must be a tree. The

probability of a tree T is

$$P(T|G) = \prod_{(u,v)\in E_T} \beta \prod_{u\in V_T,(u,v)\in E\setminus E_T} (1-\beta) \tag{3.1}$$

We require influence edges and thus all tree edges to be directed forward in time. We can include this requirement into the cascade likelihood by assigning tree edges between documents u and v with $t_u >= t_v$ probability 0. To this end, we introduce a term $P_c(u,v)$, the probability that an edge (u,v) participating in cascade c transmitted m, as

$$P_c(u,v) = \begin{cases} 1, & \text{if } t_d < t_{d'} \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

and the likelihood that cascade c spread over a given diffusion tree T as

$$P(c|T) = \prod_{(u,v)\in E_T} P_c(u,v) \tag{3.3}$$

Note that $P(c|T) = 0$ if E_T has at least one edge that goes backward in time, and $P(c|T) = 1$ otherwise. Then we can state the probability that c spread in tree pattern T in the graph G as

$$P(c,T|G) = P(c|T)P(T|G) \tag{3.4}$$

$$= \prod_{(u,v)\in E_T} \beta P_c(u,v) \prod_{u\in V_T,(u,v)\in E\setminus E_T} (1-\beta) \tag{3.5}$$

Since the nodes participating in c are known, but the diffusion tree T is not, we sum out the diffusion tree and get the likelihood of a cascade c in graph G

$$P(c|G) = \sum_{T\in\mathcal{T}_c(G)} P(c,T|G) \tag{3.6}$$

where $\mathcal{T}_c(G)$ is the set of all possible diffusion trees for cascade c in graph G. The cascades are independent, so we can state the likelihood of all cascades, i.e., the complete data likelihood given a graph G, as the product of the individual likelihoods

$$P(C|G) = \prod_{c\in C} P(c|G) \tag{3.7}$$

To find a graph G that explains the observed cascades in the best possible way, we would like to maximize Equation 3.7. Actually, Equation 3.7 resembles the likelihood formulated for the derivation of the objective function of the NETINF algorithm [12]. In the NETINF model, however, nodes do not have timestamps. Instead, contagions can infect nodes at different times,

and the data contains *hit times* for each contagion infecting a node. In the corresponding generative model of cascades, when a node u successfully transmits a contagion c to a node v, the infection time of v with c is sampled from a probability distribution (the authors investigate the exponential and the power-law distribution). Thus, the probability that a node u infects a node v with contagion c is defined accordingly as a function that decreases in the amount of time passed between the infections of u and v. However, by replacing this term with our definition of $P_c(u, v)$, the complete data likelihood stated in [12] is equivalent to Equation 3.7.

The NETINF algorithm finds a near-optimal solution of an approximate version of this likelihood function. Although the resulting graph does not necessarily have to be a directed acyclic graph with edges directed only forward in time as we require it, our definition of $P_c(u, v)$ makes sure that edges violating these properties do not yield an improvement in likelihood and are thus not chosen by the algorithm. Thus, NETINF solves our problem of document network inference based on memes.

3.3 Review of the NETINF algorithm

A brief overview of the rest of the derivation of the algorithm is given in the following: To model noise, the authors introduce the concept of *ε-edges*. The ε-edges are a set of edges E_ε with $E \cap E_\varepsilon = \emptyset$ and $E \cup E_\varepsilon = VxV$, i.e. all possible edges not present in graph G. Contagions can also travel over ε-edges, but the probability ε of an infection is much smaller than the probability β of infection over a graph edge. By this, every node can influence every other node and cascades can be explained even if there are nodes with no incoming edges from another node participating in the cascade.

The cascade and tree likelihood including the terms for ε-edges is

$$P(c, T|G) = \beta^q \varepsilon^{q'} (1-\beta)^s (1-\varepsilon)^{s'} \prod_{(u,v)\in E_T} P_c(u, v) \tag{3.8}$$

where q is the number of transmitting graph edges, q' is the number of transmitting ε-edges, s is the number of non-transmitting graph edges, and s' is the number of non-transmitting ε-edges. To simplify the problem, the authors approximate $P(c, T|G)$ as

$$P(c, T|G) \approx \beta^q \varepsilon^{q'} (1-\varepsilon)^{s+s'} \prod_{(u,v)\in E_T} P_c(u, v) \tag{3.9}$$

and replace the sum in Equation 3.6 by the maximum

$$P(c|G) \approx \max_{T\in\mathcal{T}_c(G)} P(c, T|G) \tag{3.10}$$

The final objective function is defined as the log-likelihood with the approximations 3.9 and 3.10 minus the log-likelihood for an empty graph $\overline{K}$:

$$F_C(G) = \sum_{c \in C} \max_{T \in \mathcal{T}_c(G)} \log P(c, T|G) - \max_{T \in \mathcal{T}_c(\overline{K})} \log P(c, T|G) \tag{3.11}$$

$$= \sum_{c \in C} \max_{T \in \mathcal{T}_c(G)} \sum_{(u,v) \in T} \log \beta P_c(u, v) - \log \varepsilon \tag{3.12}$$

The objective function is monotone in $|E|$, i.e., adding an edge can only increase its value. To enforce sparsity of the graph (the complete graph would achieve the maximal value), the number of chosen edges is constrained to be smaller or equal than a given parameter k. Thus, the network inference problem can be stated as follows:

Simplified diffusion network inference problem Find a graph $\hat{G}$ that satisfies

$$\hat{G} = arg \max_{G, |G| \leq k} F_C(G) \tag{3.13}$$

The objective function satisfies the *submodularity* property.

Definition 2 *For Ω a set, a function $f : 2^{\Omega} \to \mathbb{R}$ is submodular if and only if*

$$\text{For } A, B \subseteq \Omega, A \subseteq B, \text{ and every } a \in \Omega \setminus B \text{ it holds that}$$
$$f(A \cup \{a\}) - f(A) \geq f(B \cup \{a\}) - F(B)$$

Intuitively, this means that the *marginal gain* in objective function of adding an edge to the set of chosen edges earlier is never smaller than of adding it later. In general, it is NP-hard to maximize submodular functions, but the greedy algorithm achieves a guaranteed $1 - \frac{1}{e}$ approximation [22]. Consequently, the NETINF algorithm starts with an empty graph and in each iteration, picks the edge that improves $F_C(G)$ the most, until k edges are chosen. The maximum weight directed spanning trees necessary for the computation of the objective function are updated for each new edge. This is an efficient operation since to compute a spanning tree in a directed acyclic graph, one can just pick the maximum weight incoming edge for every node.

Chapter 4

Joint Inference of Concepts and Document Network

In the model described in Section 3, we assume that a document can be characterized by the set of memes it mentions. For example, the usage of the technical term *support vector machine* indicates that the corresponding classification algorithm plays a role in the document. However, a scientific paper typically mentions a large number of technical terms, many of them actually related to the same concepts. A paper about support vector machines is likely to also contain terms like *support vectors*, *kernel*, or *maximum margin*, all belonging to the *support vector* concept. In our epidemiological model where we assume that concepts diffuse from document to document, it would thus be more natural if the actual unit of contagion was not a single meme (like *support vector machines*), but rather a group of memes related to the same concept (like *support vector machine*, *support vectors*, *kernel*, and *maximum margin*). In analogy to the intuitive notion of a concept we thus define

Definition 3 *A related-meme-set is a small set of memes that describe the different components or aspects of a specific domain concept.*

In the following, we will just call related-meme-sets *concepts*. Note that a concept is different from a topic in topic-modeling in two ways: First, a concept is a set, not a distribution. Second, a concept should combine well-defined memes that are closely related, while a topic should combine words that, if used together, constitute a certain theme. Furthermore, for simplicity we model concepts as non-overlapping, i.e., each meme is associated with exactly one concept. This is clearly a simplification of reality and limits the expressiveness of our model. However, while words in general can have very different meanings in different contexts, or can even by homonyms, memes or technical terms generally have well-defined meanings. Thus, we

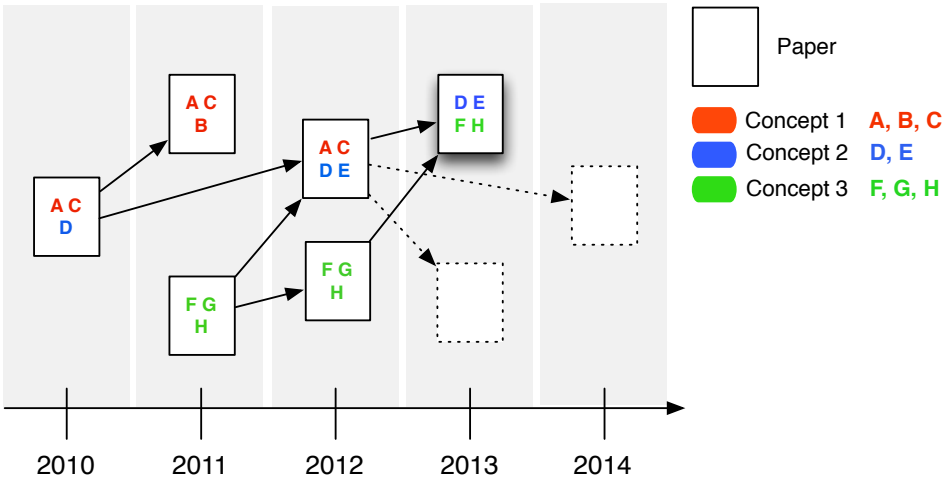


Figure 4.1: Concept diffusion. Each document's realization of a concept is a subset, and the realizations mutate in a Markov process from document to document.

believe that a model assuming that a meme can be assigned to exactly one concept is simplified but still reasonable.

In Section 4.1, we describe our model of concept diffusion and mutation in detail. Since concepts are generally not known in advance, our goal is to infer both concepts and the document network. To this end, we present a generative model of both concepts and document content in Section 4.2. Based on this model, we derive a joint objective function and state the *Joint Concept and Document Network Inference Problem* in Section 4.3.

4.1 Concept Diffusion and Mutation

In this section, we define a diffusion model based on concepts. The diffusion process is analogous to the one described in Section 3.1: We have a directed acyclic document graph G^* with edges only going forward in time. When a document is infected with a concept γ, it gets a chance to infect its neighbors. The first infection of a document with a concept determines its adoption.

The difference is the following: Since a concept is a set of memes, an author might not use all the memes if she is infected with the concept γ, but rather choose a subset $r \subseteq \gamma$, her *realization* of the concept. Furthermore, the subset she chooses might actually be similar to the subset chosen by the infecting document, since she is influenced by the infecting document. We model this process as follows: When a document d successfully infects a document d'

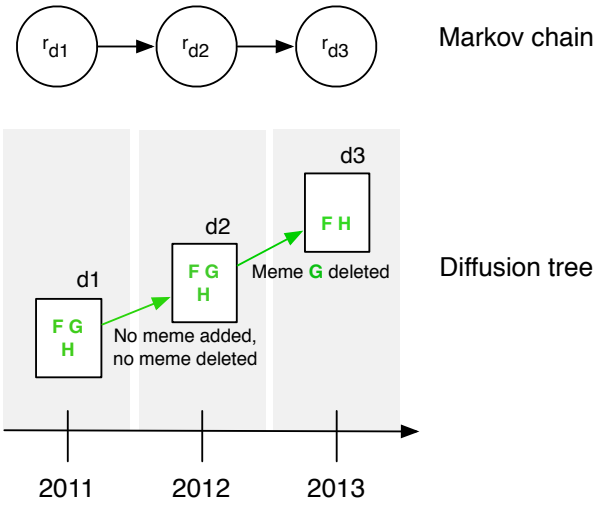


Figure 4.2: Diffusion tree corresponding to concept 3 (let us call it γ) in Figure 4.1 and corresponding Markov chain of realizations $r_{\gamma,d1}$ to $r_{\gamma,d3}$.

with a concept γ, the realization of γ in d', $r_{\gamma,d'}$ is a random variable that depends (only) on the realization of γ in d, $r_{\gamma,d}$. Thus, the realizations of a concept γ follow a *Markov process* along the corresponding diffusion tree T. For an illustration of the diffusion process, consider Figure 4.2. It shows a graph of documents whose layout is such that a document d's horizontal position corresponds to its timestamp t_d. The illustration is a snapshot at time $t = 2013$ where a new document is added. Symbols A, B, C, and so on represent memes, and the colors represent concepts. Note the small mutations, i.e., additions and deletions of memes, from one document to the next. We model these mutations as follows: Assume a document d' is infected with γ by a document d. Given the parent realization $r_{\gamma,d}$, we initialize $r_{\gamma,d'}$ with $r_{\gamma,d}$. Then, memes from γ are added as long as a biased coin with heads probability p_{add} shows heads, and memes are deleted as long as a biased coin with heads probability p_{del} shows heads. One specific diffusion tree and the corresponding Markov process is shown in Figure 4.2 (for simplicity of illustration we show a chain, but it could be a general tree). The process is formalized in Algorithm 1.

The document d introducing the concept γ does not have a parent that infected it with γ. For the initial realization $r_{\gamma,d}$, we assume that every meme in γ is sampled uniformly at random with probability p_{init}.

Based on the described model, we can formulate the likelihood of a cascade c spreading in a particular tree pattern T analogously to the case in Section 3.2. The only difference is that we now also have to attribute for the mutation. To do this, we again give an alternative definition of $P_c(d, d')$, the probability

Algorithm 1 Mutation

Input: $\gamma, r_{\gamma,d}$
Output: $r_{\gamma,d'}$
  $r_{\gamma,d'} \leftarrow r_{\gamma,d}$
  $A \leftarrow \gamma \setminus r_{\gamma,d}$
  $B \leftarrow r_{\gamma,d}$
  while $coin\_flip(p_{add}) = H$ **and** $A \neq \emptyset$ **do**
    $add \leftarrow choose\_uar(A)$
    $A \leftarrow A \setminus \{add\}$
    $r_{\gamma,d'} \leftarrow r_{\gamma,d'} \cup \{add\}$
  while $coin\_flip(p_{del}) = H$ **and** $D \neq \emptyset$ **do**
    $del \leftarrow choose\_uar(B)$
    $B \leftarrow D \setminus \{del\}$
    $r_{\gamma,d'} \leftarrow r_{\gamma,d'} \setminus \{del\}$

that cascade c spread from document d to d'. According to our model of mutation, it is

$$P_c(d,d') = \prod_{i=0}^{a-1} p_{add} \frac{1}{|A|-i} \prod_{i=0}^{b-1} p_{del} \frac{1}{|B|-i} (1-p_{add})^{\mathbf{I}_{a<|A|}} (1-p_{del})^{\mathbf{I}_{b<|B|}} \quad (4.1)$$

where the sets A and B are again the sets of memes that could have potentially been added and deleted, respectively, and a and b denote the actual number of added and deleted memes. Again, to ensure that cascades can only travel forward in time, we define $P_c(d,d') = 0$ if $t_d \geq t_{d'}$. Given a set of documents V and a set of concepts Γ, one can easily extract the realization $r_{\gamma,d}$ of a concept $\gamma \in \Gamma$ by computing the intersection $M_d \cap \gamma$. The documents participating in cascade c are all documents with a non-empty intersection with γ (i.e., containing at least one meme from γ). A cascade is now a set of documents with the corresponding realizations. With the alternative definition of $P_c(d,d')$, NETINF also solves the problem of inferring a document network based on cascades of mutating concepts.

4.2 Joint Model for Concept Generation and Diffusion

In general, we might not know the concepts in advance. Concepts could be constructed manually by domain experts or automatically by using clustering or topic modeling methods. Since concepts travel over a network of documents, however, a method to infer concepts could profit from network information. Thus, in the following we address the problem of jointly inferring concepts and the document network. To this end, we define a generative model for both concepts and documents.

We would like to allow concepts that are wide-spread (i.e., have large cascades), more memes than concepts that only occur in few documents, since concepts naturally evolve in the diffusion process and more and more aspects of a concept appear as more documents adopt it. To this end, we include the generation of concepts into our generative model for documents: First, we define an *abstract concept* whose meme set is not yet defined. This abstract concept spreads over the document graph $G^* = (V, E)$ according to the independent cascade model for documents as described in Section 3.1. The result of this process is a diffusion tree $T = (V_T, E_T)$ with $V_T \subseteq V$ and $E_T \subseteq E$. Now, the corresponding *concrete concept* γ is constructed as a random set of memes whose size n is sampled from a normal distribution with mean proportional to the tree size.

$$n \sim \mathcal{N}(\eta|T|, \sigma^2) \tag{4.2}$$

where η and σ are parameters of the model. Finally, the realizations $r_{\gamma,d}$ are created by diffusing γ along the fixed diffusion tree T, where mutations happen as described in Section 4.1.

4.3 Joint Objective Function and Problem Statement

As a first step towards our goal of recovering concepts and diffusion graph from observed documents, we derive a joint objective function. We use the symbols $T = (V_T, E_T)$ for the diffusion tree of a concept γ, and R for the realizations of γ in the documents that participate in the cascade. According to the generative model described above, we can state the probability of a specific concept γ spreading over a tree T and producing realizations R as

$$P(T, \gamma, R|G) = \underbrace{P(T|G)}_{\text{Tree}} \underbrace{P(\gamma|V_T)}_{\text{Concept size}} \underbrace{P(R|\gamma, T)}_{\text{Realizations}} \tag{4.3}$$

In analogy to the derivation in Section 3.2, the probability that the diffusion tree T arises in graph G is

$$P(T|G) = \prod_{(d,d')\in E_T} \beta \prod_{d\in V_T,(d,d')\in E\setminus E_T} (1-\beta) \tag{4.4}$$

We defined the concept size to be normally distributed around a mean proportional to the tree size. So the probability of concept γ given diffusion tree T is simply

$$P(\gamma|V_T) = \mathcal{N}(|\gamma|; \eta|V_T|, \sigma^2) \tag{4.5}$$

Finally, the mutations along the diffusion tree follow a Markov process, so the probability of the realizations along the tree T is the product of all mutation probabilities.

$$P(R|\gamma, T) = P(T|G) = \prod_{(d,d')\in E_T} P_c(d, d') \tag{4.6}$$

The size of γ does not depend on the shape of T, but just on its node set. Given the data, i.e., documents d and the associated sets of memes M_d, for a fixed concept γ, we can observe the cascade $c = (V_T, R)$. As described above, it contains all documents that contain at least one meme from γ. Thus, we have $P(\gamma|V_T) = P(\gamma|c)$. To get the probability of the concept and the realizations, we sum out the unobserved trees, where we pull $P(\gamma|c)$ which does not depend on the tree shape out of the sum.

$$P(\gamma, R|G) = P(\gamma|c) \underbrace{\sum_{T \in \mathcal{T}_c} P(T|G)P(R|\gamma, T)}_{\text{NETINF cascade likelihood}} \tag{4.7}$$

Note that the sum in Equation 4.7 is just the cascade likelihood for the cascade induced by concept γ as derived in Section 3.2, which arises because we use the same diffusion process split into two parts (first the diffusion tree is created, and then the concepts diffuse over it). Since we added only a multiplicative term that is independent of the graph G, ε-edges and approximations can be introduced as described in Section 3.2. Finally, when the individual likelihoods are multiplied and the logarithm is taken, we arrive at our objective function

$$F'(G, \Gamma) = F_C(G) + \sum_{\gamma \in \Gamma} \log P(\gamma|c_\gamma) \tag{4.8}$$

where $F_C(G)$ is the NETINF objective function stated in Equation 3.12 on page 18. Thus, we formally define the joint concept and document network inference problem as follows:

Joint Concept and Document Network Inference Problem For a given set of documents V, timestamps t_d, and meme sets M_d, find a graph $\hat{G}$ and a concept set $\hat{\Gamma}$

$$\hat{G}, \hat{\Gamma} = arg \max_{|G| \leq k, |\Gamma| \geq y} F'(G, \Gamma) \tag{4.9}$$

where k (maximal number of edges) and y (minimal number of clusters) are given parameters.

It is not clear that the objective function can always be increased by decreasing the number of concepts. For fewer concepts, fewer edge transmissions and potentially fewer ε-edges have to be explained, but in return, the amount of mutation is likely to increase. We leave it as an open question to theoretically analyze this interaction. Empirically, we observed that the objective function favors a small number of clusters, and thus put a lower bound y on the number of clusters. To enforce that the minimal number of clusters y is always attained, one can subtract a constant penalty for every cluster from the objective function.

Chapter 5

The DocNETINF Algorithm

In this section, we present DocNETINF, the algorithm we developed to solve the joint concept and network inference problem. For fixed Γ, the objective function in Equation 4.8 reduces to the NETINF objective function plus a constant term. Consequently, the objective function seems to be at least as hard to maximize as the original NETINF objective function, which is NP-complete. An efficient algorithm that finds an exact solution seems out of reach. Thus, to find a local optimum, we take the approach of alternately maximizing w.r.t. G for fixed Γ and Γ for fixed G. Although the algorithm is a heuristic, we show that it can successfully recover concepts and a document network by experiments on simulated data (cf. Section 6).

Optimizing for Γ when G is fixed is a clustering task, i.e., memes are clustered to concepts. In the following, we first describe the clustering subroutine of DocNETINF and finally state the complete algorithm.

5.1 Meme Clustering

The clustering algorithm needs to evaluate the objective function for a given clustering. Since it decomposes into a sum of the individual concepts, each contribution can be computed independently. To this end, for a concept γ, its cascade c, i.e., all documents in which γ occurs, have to be extracted. This can be done efficiently by using a prepared hash-table that maps each meme to the set of all documents that contain it. Since concepts are non-overlapping and one occurrence of a meme belonging to a concept already counts for an occurrence of the concept, the cascade set is just the union of all the sets with concept memes as keys. This immediately yields the cascade size, thus the term $P(\gamma|c)$ is easy to compute. The contribution of a cascade c to the NETINF objective function boils down to the following

quantity:

$$\max_{T \in \mathcal{T}_c} \sum_{(d,d') \in E_T} \log \beta P_c(d,d') + (|c| - 1 - |E_T|) \log \varepsilon \tag{5.1}$$

This can be interpreted as the log-probabilities of transmission for every present edge, and an ε-penalty for every missing edge (if T is a full tree, i.e., each node except the root has a parent, it has $|c| - 1$ edges). The probabilities $P_c(d,d')$ can be computed based on the realizations $r_{\gamma,d}$ and $r_{\gamma,d'}$ with the formula given in Equation 4.1 on page 22. To compute the most likely tree, we use the observation made in [12] that the tree we are looking for is just the maximum weight directed spanning tree of the nodes belonging to c, where the weight $w_{d,d'}$ of an edge (d,d') is defined as

$$w_{d,d'} = \begin{cases} \max(\log \beta P_c(d,d'), \log \varepsilon), & if (d,d') \in E_G \\ \log \varepsilon, & \text{otherwise} \end{cases} \tag{5.2}$$

In a directed acyclic graph (G is a DAG since edges can only go forward in time), this tree can be computed by just choosing the incoming edge with maximum weight for each node in c except the root. Thus, for each cascade node d, we iterate over all incoming edges from other cascade nodes d', compute the corresponding edge weight and add the maximum of the weights to the objective function.

Agglomerative Clustering Algorithm To do the clustering based on the inferred graph G, we use an agglomerative clustering algorithm: The clusters are initialized to memes, and each cluster's contribution to the objective function is computed. We then evaluate the change in the objective function value for all possible merges of two clusters by subtracting the individual contributions from the potential contribution of the merged cluster. All changes that would improve the objective function value are inserted into a priority queue, where highest improvement has highest priority. Additionally, for each cluster we keep a pointer to the changes in the priority queue it would participate in.

We then iteratively retrieve the change with highest objective function increase from the priority queue. We perform it, i.e., update the clusters datastructure and objective function value and remove all changes associated with one of the participating clusters (which now do not exist any more) from the priority queue. Finally, we consider all possible merges involving the new cluster and again insert the ones that increase the objective function into the priority queue. We stop when we have reduced the number of clusters to a given value. The algorithm is outlined formally in Algorithm 2.

The asymptotic complexity of this algorithm can be analyzed as follows: Since cascades are generally small, we assume that computing the objec-

tive function for a given cascade takes constant time. For n, the number of memes, the initial all-pairs objective function change computation is in $O(n^2)$. Inserting a change into the priority queue is in $O(\log(n^2)) = O(\log n)$ since the number of changes can never be larger than n^2 and insertion is logarithmic in the number of elements in the queue (we use a balanced tree structure). Since each change reduces the number of clusters by 1, there can maximally be n iterations. In each iteration, $O(n)$ changes are removed and inserted into the priority queue, each operation taking $O(\log n)$. Thus, the asymptotic complexity is $O(n^2 \log n + n^2 \log n) = O(n^2 \log n)$. A simple heuristic we use to speed up the execution in practice is to only consider merges of clusters that cooccur at at least one document.

Algorithm 2 Agglomerative meme clustering algorithm

Input: document meme sets M_d and timestamps t_d for documents $d \in V$, number of concepts y
Output: concepts Γ

```
Γ ← ∅
for all m ∈ V do
    Γ ← Γ ∪ {({m}, compute_obj_func({m})) }
change_queue ← new ChangeQueue()
for all ((γ, v), (γ', v')) ∈ Γ × Γ do
    v'' ←compute_obj_func(γ ∪ γ')
    if v'' − v − v' > 0 then
        change_queue.insert(v'' − v − v', (γ, γ'))
while |Γ| > y and not change_queue.is_empty() do
    v, (γ', γ'') ← change_queue.get_best()
    Γ ← Γ \ {(γ', v_γ'), (γ'', v_γ'')}
    change_queue.remove_all({γ', γ''})
    γ ← γ' ∪ γ''
    for all (γ', v') ∈ Γ do
        v'' ← compute_obj_func(γ ∪ γ')
        if v'' − v − v' > 0 then
            change_queue.insert(v'' − v − v', (γ, γ'))
    Γ ← Γ ∪ {(γ, v)}
```

5.2 Alternating Algorithm

As mentioned above, when $\hat{\Gamma}$ is fixed, maximizing $F'(G, \Gamma)$ is equivalent to maximizing $F_C(G)$. Thus, we use NETINF as a subroutine of the algorithm: In the *network inference* step, cascades are extracted from the documents based on the current concepts $\hat{\Gamma}$ and a new graph $\hat{G}$ is inferred. When the

Algorithm 3 DocNETINF

Input: documents d, timestamps t_d, and meme sets M_d
Output: graph G and concepts Γ
  $\Gamma \leftarrow$ memes
  while objective function increased by at least 1% **do**
    $G \leftarrow$NETINF with cascades induced by Γ ▷ Network inference step
    $\Gamma \leftarrow$Cluster memes based on G ▷ Concept step

graph $\hat{G}$ is fixed, we use the clustering algorithm described above to find a new clustering $\hat{\Gamma}'$ that achieves a high value of $F'(G, \Gamma)$. The procedure is outlined in Algorithm 3.

To prove the convergence of DocNETINF, we would have to argue that the objective function value increases monotonically over the iterations, i.e., the value can only stay constant or increase in each step. Basically, if we have a graph $\hat{G}$, we fix $\hat{\Gamma}$, and infer a new graph $\hat{G}'$, we know that there exists a graph that does not make the objective function decrease, namely $\hat{G}' = \hat{G}$. If the network inference step would be guaranteed to find the $\hat{G}'$ that maximizes $F_C(G)$, this solution would be at least as good as $\hat{G}$ (otherwise $\hat{G}$ would be a better solution and $\hat{G}'$ would not maximize the objective function). However, the NETINF algorithm is guaranteed to find a $1 - \frac{1}{e}$ approximation of the optimum, but not the optimal solution. So it could happen that a new solution $\hat{G}'$ is actually worse than the previous solution $\hat{G}$. However, the authors of NETINF show empirically that the solution found by NETINF is very close to the optimal in practice. Thus, we do not expect the objective function value to decrease significantly from one network inference step to the next.

Similarly, for a new fixed $\hat{G}$, the last clustering solution $\hat{\Gamma}$ would also be a valid new solution. However, the clustering algorithm is greedy and does not have any performance guarantees. Thus, a new clustering solution $\hat{\Gamma}'$ found based on a new graph $\hat{G}'$ does not necessarily have to be better than the previous solution $\hat{\Gamma}$. Nevertheless, in our experiments on simulated data, we observed a monotonic increase of the objective function over the iterations (cf. Section 6).

Chapter 6

Experimental Evaluation of DocNETINF on Simulated Data

In this section, we present the evaluation of the DocNETINF algorithm on simulated data. To generate ground-truth data, we simulated the diffusion process of concepts over a network of documents. We ran DocNETINF on the resulting synthetic documents and compared the inferred network and concepts to the ground-truth network and the ground-truth concepts. In Section 6.1, we describe the data simulation process. The metrics we use to measure the quality of the results are introduced in Section 6.2. In Section 6.3, we describe the baseline methods against which we compare DocNETINF. Finally, we present the results of the evaluation on simulated data in Section 6.4.

6.1 Data Simulation

As a first step in our data simulation process according to the concept diffusion model described in Section 4.2 we need a set of documents V, associated timestamps t_d, and a directed acyclic graph $G^* = (V, E)$, where edges only point forward in time, i.e., for each edge $(d, d') \in E$ it holds that $t_d < t'_d$. A natural choice for such a graph would be (a subgraph of) a real citation graph, since we expect an influence-graph as we would like to detect it to have similar structure as a citation graph. The scientific paper data at our disposal (cf. Section 7.4.1) contains timestamps and citations, but since citation graphs are sparse, it rendered difficult to extract sufficiently dense paper graphs of reasonable size for our experiments (roughly 500 nodes).

Thus, we created random graphs according to the *linear growth copying model* [16] for our experiments. The linear growth copying model is a random graph model originally designed to imitate properties of the web graph, where nodes are webpages and edges are links between them. In particular,

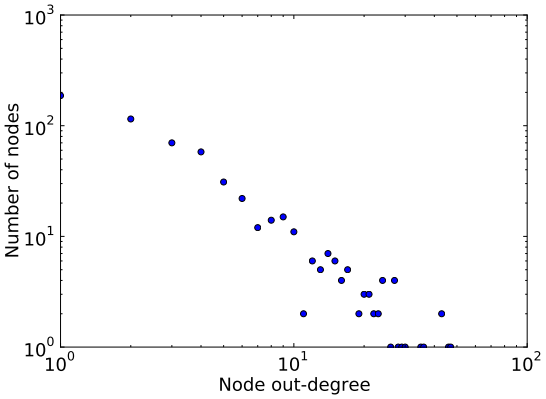
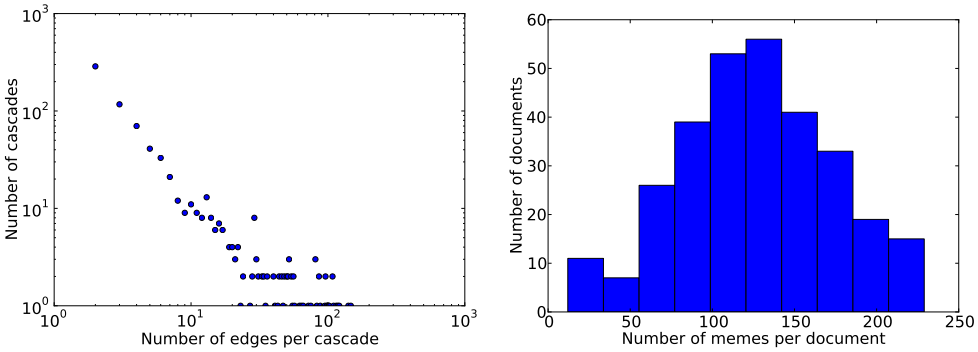


Figure 6.1: Out-degrees of a graph generated according to the adjusted linear growth copying model with 1000 nodes, $\alpha = 0.5$, $d = 3$, and timestamps from 1980 to 2010.

a power-law distribution of the number of in-links naturally arises from the random creation process. The process is as follows: The parameters are a copy factor $0 \leq \alpha \leq 1$ and a constant out-degree d. The graph creation is simulated over time. For a new node with timestamp t_d, a node p is chosen from the already-existent nodes, so $t_p < t_d$. With probability α, the i-th out-link of d is chosen to be the i-th outlink of p (the link is *copied*), and with probability $1 - \alpha$, its head is chosen uniformly at random from the existent nodes. The time granularity we consider is years, since the papers in are dataset are timestamped with the publication year. As in the real data set, we would like to have multiple papers in the same year. So, deviating from the original description of the model, for every timestamp t, we create a bunch of nodes according to the process described above, and then insert them into the graph all at once. Finally, we reverse all edges to get an influence graph from a link graph. For our experiments, we fixed $\alpha = 0.5$, $d = 3$, and timestamps from 1980 to 2010 (with an equal amount of nodes for each timestamp). Figure 6.1 shows a log-log plot of the out-degree distribution of a 1000 node graph created according to the described process. As observable from the plot, the out-degree distribution follows a power-law.

After this process, we have a document graph G^* of empty timestamped documents. To create the document content, we simulate the concept diffusion model as described in Section 4.2. In particular, for a fixed number of times, we create an abstract concept (recall that an abstract concept is a concept that does not have memes yet), place it at a node chosen uniformly at random, and execute the random diffusion process from there to get a



(a) Cascades per edge (b) Memes per document

Figure 6.2: Results of diffusing 750 concepts over a 300 node random copy graph, $\beta = 0.5$, $\eta = 0.15$, and $\sigma^2 = 3$.

diffusion tree T. Then, we sample the concept size from a Gaussian distribution with mean proportional to the tree size (we round to get integer sizes), and a random set of memes of that size. Finally, we sample the realizations of the tree nodes according to the mutation model given in algorithm 1 on page 22. At the end of this process, a document's content is collected as the union of all concept realizations sampled for the document. Figure 6.2 shows histograms of the cascade sizes (in log-log scale) and number of memes per document that result from the simulation of 750 concepts on a 300 node random copy graph with parameter $\beta = 0.5$, $\eta = 0.15$, and $\sigma^2 = 3$. The mean concept size is 2.3 memes.

6.2 Metrics

To assess the performance of the algorithm on the simulated data, we use three metrics for the three objects of interest: the inferred graph G, the set of inferred concepts Γ, and the inferred diffusion trees T. They are briefly described in the following.

Graph agreement To compare the inferred graph $G = (V, E_G)$ with the ground-truth graph $G^* = (V, E)$, we use precision and recall of the edge set E_G w.r.t. the edge set E. Precision and recall are defined as usual as

$$\text{Precision} = \frac{|E \cap E_G|}{|E_G|} \qquad \text{Recall} = \frac{|E \cap E_G|}{|E|} \tag{6.1}$$

NETINF greedily optimizes the objective function, so edges with higher gain are chosen earlier. Thus, we can construct a precision-recall curve by adding the edges returned by NETINF to the inferred graph G one by one. To

compress this curve to a single number, we compute the *area under the curve* of the precision-recall curve AUC-PR.

Concept agreement The concept inference is a clustering task on memes. A standard metric for evaluating a clustering w.r.t. a ground truth clustering is *Rand index*, which is defined as

$$\text{Rand index} = \frac{a+b}{\binom{n}{2}} \tag{6.2}$$

with a the number of pairs that are correctly put into the same cluster, and b the number of pairs that are correctly put into different clusters. Since we have many small clusters, b is very large even for bad solutions, and the rand index is always close to 1. Thus, we consider the clustering task as a classification task of pairs of elements, and compute precision and recall of pairs in the same cluster. Formally, for sets of pairs P_Γ (inferred pairs) and P_{Γ^*} (ground truth pairs), we have

$$\text{Precision} = \frac{|P_\Gamma| \cap |P_{\Gamma^*}|}{|P_\Gamma|} \qquad \text{Recall} = \frac{|P_\Gamma| \cap |P_{\Gamma^*}|}{|P_{\Gamma^*}|} \tag{6.3}$$

As a combined metric we consider the harmonic mean of precision and recall or f-measure

$$\text{F-measure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

Diffusion tree agreement The results of the algorithm are a document graph G and a clustering of memes into concepts Γ. For each concept $\gamma \in \Gamma$ we can additionally compute the most likely diffusion tree as described in Section 5.1. Since in practice, one might be interested in the concrete diffusion tree of a given concept, we consider as an additional metric the agreement of the inferred diffusion trees with the ground truth diffusion trees. However, each diffusion tree belongs to a concept, and since the inferred concepts do not perfectly match the ground truth concepts, it is not obvious which inferred tree to compare to which ground-truth tree. To define a metric, we think of the following hypothetical interaction of a user with a system implementing the algorithm:

- A user chooses a meme m of interest and wants to know the diffusion tree T^* on G^* of the concept γ^* the meme belongs to.
- The system chooses the concept the meme is inferred to belong to, computes the corresponding most likely diffusion tree T for this concept on the inferred graph G and returns it.

We define the *diffusion tree agreement* metric as the expected Jaccard similarity of the edge sets E_T (the returned diffusion tree) and E_{T*} (the ground truth diffusion tree), where the expectation is over all memes in the meme vocabulary M. Formally,

$$\text{Diffusion Tree Agreement} = \frac{1}{|M|} \sum_{m \in M} \frac{|E_{T_m^*} \cap E_{T_m}|}{|E_{T_m^*} \cup E_{T_m}|} \tag{6.5}$$

6.3 Baseline Methods and Upper Bound

To the best of our knowledge, the task of jointly inferring concepts and a network of documents has not been addressed in the literature before. Thus, we compare our method to the following two simple baselines.

NETINF with memes as contagions We assume each meme to be one independent concept and run NETINF (modified to use the appropriate $P_c(d, d')$ for documents described in Section 3.2) to get an output graph G. Note that this corresponds to the first iteration of DocNETINF. There is no notion of a concept consisting of more than one meme in this baseline method.

Hierarchical clustering of memes + NETINF with mutation Memes that belong to the same concept travel over the same diffusion tree. Since there is mutation, the set of hit documents is not the same for all memes in the concept. However, memes belonging to the same concept still often cooccur, which corresponds to the intuitive expectation. This motivates the following baseline method: We first try to find the concepts with a standard hierarchical clustering based on a dissimilarity matrix of memes. To this end, we define the document sets

$$D_m = \{d \in D | m \in M_d\} \tag{6.6}$$

and the dissimilarity of two memes m and m' based on the Jaccard similarity of the document sets as

$$D(m, m') = 1 - \frac{|D_m \cap D_{m'}|}{|D_m \cup D_{m'}|} \tag{6.7}$$

As a linkage method, we use *average linkage*, meaning that when two clusters are combined, the distances to all other clusters are recomputed as the average distance of the contained items to each other. From the resulting cluster hierarchy, y flat clusters are formed, where y is the number of ground truth clusters. These constitute the concept set Γ, based on which we run NETINF with mutation as described in Section 4.1.

Graph	Number of nodes	300
	Node time range	1980 to 2010
	Number of edges per node	3
	Copy factor α	0.5
Diffusion model	Number of concepts	750
	Transmission probability β	0.5
	Probability of infection without edge ε	1e-10
	Concept memes per diffusion tree node η	0.15
	Variance of memes per diffusion tree node σ^2	3
	p_{init}	0.66
	p_{add}, p_{del}	0.3

Table 6.1: Default parameters for conducted experiments.

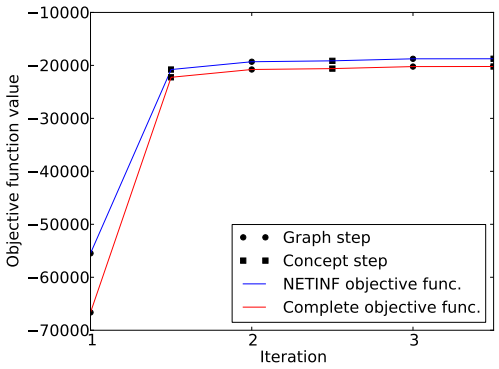
Upper Bound We also compare our algorithm to a method that is given the true concepts in advance, and only infers the graph and diffusion trees using NETINF. Note that we assume that we do not have the true concepts. However, it is interesting to compare the performance of this more informed method to DocNETINF. In particular, its performance with respect to the recovery of the ground-truth graph and the ground-truth diffusion trees is an upper bound for the performance of DocNETINF.

6.4 Results

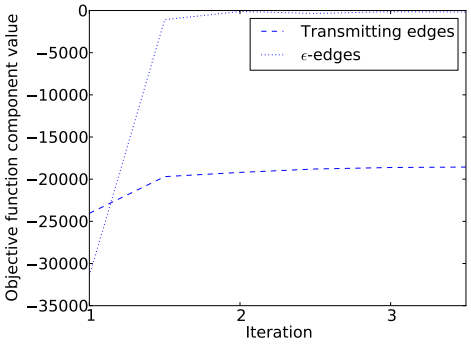
We conducted experiments to assess the performance of DocNETINF for various parameter settings. The default parameters we used are given in table 6.1. Deviations from the default values are stated at the corresponding points. In the following, we first discuss the convergence behavior of the algorithm and then show results for the ground truth recovery.

6.4.1 Convergence Behavior

Figure 6.3a shows the development of the objective function over the iterations. Note that we consider one iteration to be a graph (NETINF) and a concept (clustering) step. The value after the graph step for iteration i is shown at x-coordinate i, and the value after the concept step is shown at x-coordinate $i + 0.5$. The main observations are that there is a big jump in the value from the first graph step to the first concept step. This is due to the fact that we start with the concepts initialized to memes. First, this is a very high number of concepts and we have to pay for transmissions and ε-edges for each concept. And second, even very long cascades correspond to concepts with a single meme, which we have to pay for in the $P(\gamma|c_\gamma)$ term. The second observation is that the objective function flattens out in



(a) Objective function.



(b) Gain for transmitting and ε-edges.

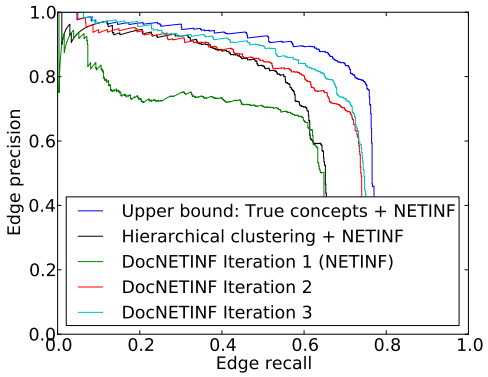
	Iteration 1 to 2	Iteration 2 to 3
Graph	41.4%	25.8%
Concepts	32.0%	20.8%

(c) Amount of change over iterations.

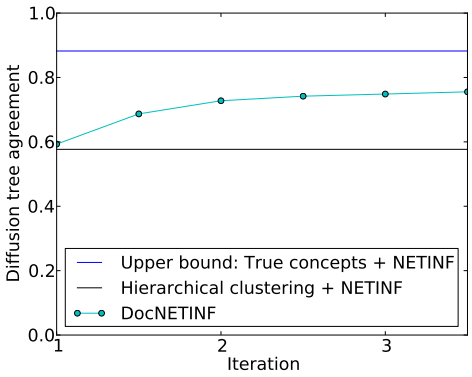
Figure 6.3: Objective function and amount of change over iterations.

iteration 3, which we generally observed in the experiments on simulated data.

In Figure 6.3b, the components of the objective function for regular transmissions and ε-transmissions are shown (the sum of the two curves is the blue line in Figure 6.3a). This shows that the big first improvement comes from a reduction in the number of ε-edges due to the fact that an ε transmission only counts once per concept. Furthermore, wrong ε edges that arise because NETINF tries to explain each meme occurrence are reduced. The likelihood of the transmissions over the graph increases as well, as similarly as described above, by reducing the number of concepts, transmissions of several memes belonging to the same concept only have to be explained once. Note that the mutation probabilities generally decrease when the size of a concept grows (the larger the concept, the more possibilities to mutate to a different realization). However, as there is a penalty for mutation in the objective function, the algorithm tries to keep the amount of mutation as likely as possible. In Table 6.3c, we give the percentage of change of the graph edge set and the concept set (in analogy to our concept agreement metric, to compute the amount of change of the concept set from one iteration to the next, the sets of pairs of memes in the same concept are considered). The amount of change decreases, but does not completely vanish although the objective function flattens out.



(a) Graph agreement.



(b) Diffusion tree agreement.

	Iteration 1	Iteration 2	Iteration 3
Concept agreement	0.756	0.797	0.797

(c) Concept agreement.

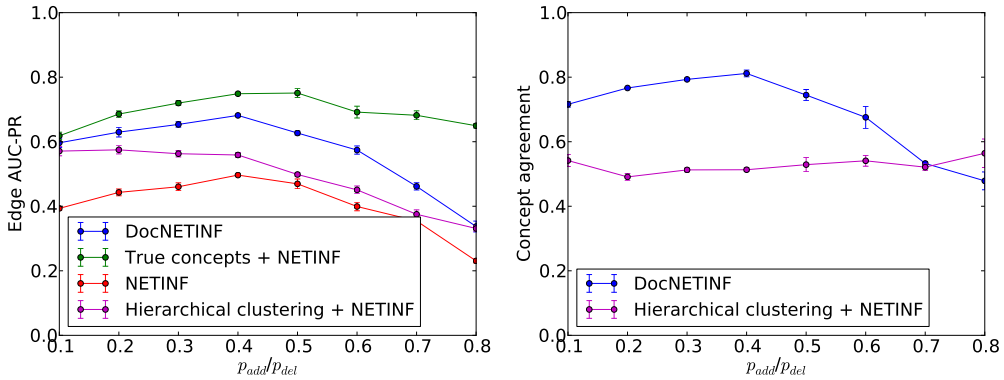
Figure 6.4: Agreement with ground truth graph, diffusion trees, and concepts.

6.4.2 Recovery of Ground Truth

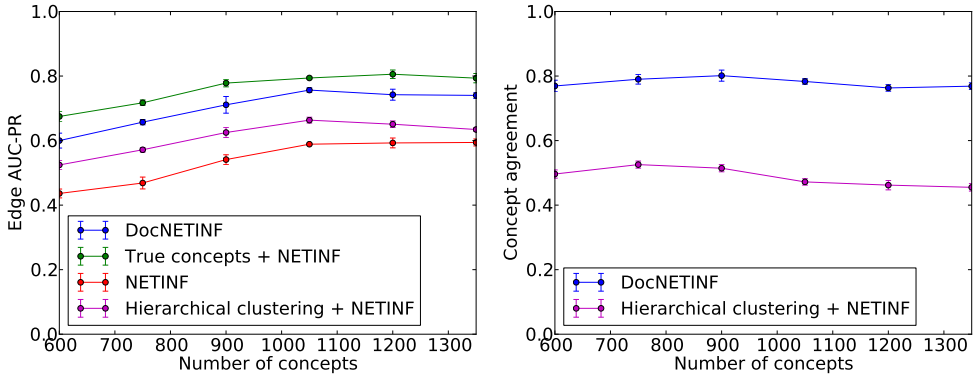
Figure 6.4a shows the precision-recall curves for the inferred edge set with respect to the ground truth graph. One can see that the performance improves over the iterations. In particular, there is a big jump from the first iteration (where concepts are just memes) to the second (the first iteration which a real estimate of concepts). In the third iteration, the precision-recall curve of DocNETINF comes close to the upper bound. Both baseline methods are clearly outperformed.

Figure 6.4b shows the agreement of the inferred diffusion trees with the ground truth trees. Note that in contrast to the graph in Figure 6.4a, the iterations are on the x-axis, and on the y-axis we show the value of the metric described in Section 6.2. Since the Hierarchical clustering + NETINF baseline and the upper bound do not iteratively improve the diffusion tree agreement, their performances are shown as constant lines. DocNETINF clearly outperforms the baseline method.

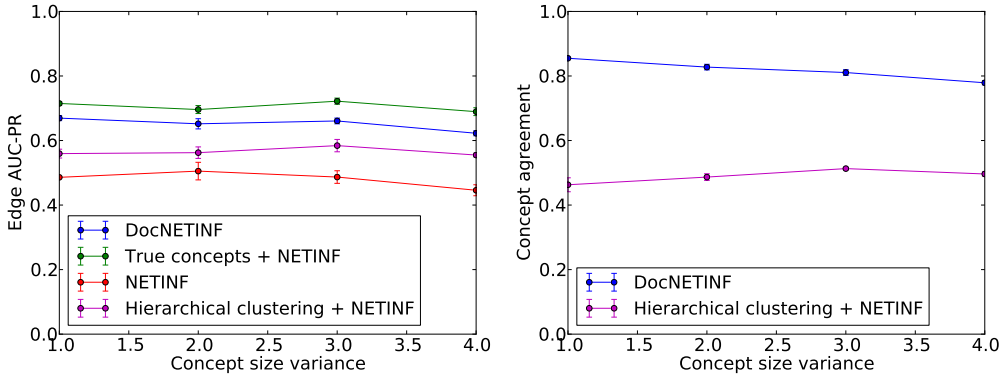
Dependency on amount of mutation The critical parameters that determine the amount of mutation are the probabilities p_{add} and p_{del} of adding/deleting a meme to/from the realization from one document to the next. Intuitively, if the amount of mutation is small, the realizations of a concept are consistent and it should be easy to infer the concept from them. As the amount of mutation increases, we expect that the identity of a concept becomes less obvious which make the problem harder. The plots shown in



(a) Graph and concept agreement for varied mutation probabilities $p_{add} = p_{del}$.



(b) Graph and concept agreement for varied number of concepts.



(c) Graph and concept agreement for varied concept size variance σ^2.

Figure 6.5: Ground truth graph and concept recovery of DocNETINF and baselines for varied parameters.

Figure 6.5a show the performance of DocNETINF with respect to the graph and concept agreement metrics in comparison to the baseline methods and the upper bound.

Generally, one can observe that our method is always better than the baselines and relatively close to the upper bound. Furthermore, the performance in terms of all metrics decreases as the mutation probabilities become large. Interestingly, however, DocNETINF and the upper bound seem to be best for mutation probabilities $p_{add} = p_{del} \approx 0.4$, and slightly worse for mutation probabilities below this value. This can be explained as follows: Since we have no time probabilities in our diffusion model, the amount of mutation between two realizations $r_{\gamma,d}$ and $r_{\gamma,d'}$ of a concept in documents d and d' with $t_d < t_{d'}$ determines the likelihood of a transmission of γ from d to d'. If the amount of mutation is very small, it is difficult to decide for a parent of d' in the diffusion tree of γ, since all potential parents have rather similar realizations of γ. If there is a certain probability for mutation in every diffusion step, however, the amount of mutation can act as a kind of *timer* or *step counter* for the diffusion process, and it gets easier to choose the right parent. Thus, although a very high mutation probability makes the problem difficult, almost no mutation is also not optimal, and the performance is best in the middle ground.

Dependency on the number of concepts The number of concepts or, equivalently, cascades that are simulated influence the performance of DocNETINF in two ways: On the one hand, we expect the clustering to become more difficult, since the density of diffusion trees and the amount of overlap of different concepts at documents increases which could cause confusion. On the other hand, the network inference becomes easier since more diffusion data is available to the algorithm. Figure 6.5b shows graph and concept agreement for a varied number of concepts. One can see that all methods' performance in terms of graph recovery increases as the number of concepts increase. Surprisingly, the concept agreement for DocNETINF is roughly constant, while the concept agreement for the Hierarchical clustering + NETINF baseline method has a slightly negative trend. We conclude that even a relatively large number of concepts seems to be distinguishable based on the result of the diffusion process, where DocNETINF has the additional advantage that the initial network inference becomes better with more concepts (cf. NETINF baseline).

Dependency on the variance of the concept sizes Recall that the concept sizes are sampled from a normal distribution with mean proportional to the cascade size (cf. Section 4.2). The corresponding variance σ^2 determines how tight cascade and concept sizes are bound together. The plots in Figure 6.5c show that the concept agreement decreases with increasing concept size

variance. However, the decrease is relatively small which suggests that the clustering does not depend heavily on the additionally term $P(\gamma|c_\gamma)$ in the objective function (cf. Section 4.3). Furthermore, the graph agreement stays roughly constant, i.e., the network inference step is not extremely sensitive to the quality of the clustering.

Chapter 7

Application to Real Data

To test our algorithm on real data, we inferred document networks and concepts for scientific papers from various Computer Science conferences. Furthermore, to show that the results are useful, we built a web application that allows a user to inspect and explore them. In Section 7.1, we describe our method to extract technical terms from document text. Then, we describe the problems that arose and our solution attempts when we applied DocNETINF to real data in Section 7.2. The demo application is presented in Section 7.3. Finally, we present results on real data and an anecdotal analysis in Section 7.4.

7.1 Technical Term Extraction

As mentioned before, the contagions we consider are memes, traces of ideas and concepts. For the scientific literature, we consider *technical terms* as such memes. The open dictionary *Wiktionary*[1] defines a technical term as *a word that has a specific meaning within a specific field of expertise*. While this is a very general definition, *specific meaning* can be interpreted as the association with a specific domain concept, occurrences of which we would like to detect as contagions with (an aspect of) the concept.

As we do not want to rely on keywords supplied for a paper or any manual preprocessing, we need a method to extract technical terms from the plain texts of papers. Several methods for *automatic term recognition* have been proposed in the literature (cf. [30] for a review and evaluation), but most of them focus on the total term frequency of a term in the corpus as an indicator for the *termhood* of a term, which is not appropriate for our application, since we would not only like to capture the most established technical terms of a domain but rather any traces of technical terminology. Furthermore,

[1] en.wiktionary.org/

many rely on parts-of-speech tagging, i.e., the assignment of word types to extracted tokens, and define filters for technical terms only allowing certain sequences of word types (e.g. *noun phrases* consisting only of nouns), which we wanted to avoid for simplicity and for the sake of not missing technical terminology that might not match these patterns.

We extract multi-word technical terms as follows: We first tokenize the plain text of a paper and stem the tokens[2]. Then, we slide windows of two, and three tokens over the list of tokens. Since we do not use a linguistic filter, the resulting n-grams may be any sequence of words. However, n-grams starting or ending with articles or prepositions are obviously not technical terms. To filter out these n-grams, we use a simple heuristic: We discard all n-grams that start or end with a stopword, where we use a standard stopword list to recognize stopwords. The remaining n-grams of tokens arising from this process are candidates and we collect their occurrence frequencies in the paper. The frequency of occurrence is basically a good indicator for the termhood of a candidate as for example argued in [30]. However, note that any shorter n-gram t nested in a longer n-gram t' has at least the frequency of occurrence of t'. Thus, we define the *effective frequency of occurrence* of an n-gram t in a document d as

$$\bar{tf}_{t,d} = \begin{cases} tf_{t,d} - \max_{t',t' \text{ contains } t} tf_{t',d} & \text{, if } \max_{t',t' \text{ contains } t} tf_{t',d} > 1 \\ tf_{t,d} & \text{, otherwise} \end{cases} \tag{7.1}$$

The rational behind this definition is the following: We do not want to count occurrences of n-grams that are always a part of another n-gram, since the actual unit of occurrence is the larger n-gram in this case. If an n-gram occurs frequently as a part of different larger n-grams, however, it is likely to be a technical term itself. Then, we count all occurrences except the ones as part of the most frequent larger n-gram.

To choose technical terms from our candidates, we follow a simple intuition: While a technical term does not necessarily have to be very frequent in the document collection, it is likely to have a high effective frequency of occurrence in at least one paper. Furthermore, we require a technical term t to have a certain proliferation and thus at least a certain document frequency df_t. On the other hand, if a term occurs in a large fraction of the documents, it is the equivalent of a stopword in the technical terminology and we also do not want to extract it. Thus, we select the set of multi-word terms as

$$T = \{t | \max_d \bar{tf}_{t,d} > \tau \wedge df_{min} \leq df_t \leq df_{max}\} \tag{7.2}$$

[2]We also keep the original tokens for the final result.

where τ, df_{min} and df_{max} are parameters of the system. We used $\tau = 5$, $df_{min} = 5$ and $df_{max} = 100$ for smaller datasets (2000 documents or less) and $\tau = 5$, $df_{min} = 10$ and $df_{max} = 250$ for larger datasets (more than 2000 nodes).

When we construct the sets of technical terms representing a document, we extract all bigrams and trigrams from the document that are an element of T. We also experimented with requiring a minimal term frequency tf_{min} in the document to include a technical term into a document's set. However, this did not yield better results and thus we set tf_{min}=1. For unigrams, the method would not yield satisfactory results, since single standard English words are likely to occur often in some document without being technical terms. The extensive use of a word in a scientific paper can, however, suggest that the corresponding concept plays an important technical role in the paper. To also extract these words from papers, we utilize a list of English words and their frequency in non-technical English texts[3]. We then compare the normalized term frequency of each word in a document with the normalized term frequency of the word in the standard English list. If the ratio is larger than a certain value (we use 50,000), we include the word in the technical term set of the document.

7.2 Adaptations of DocNETINF for Real Data

When we first applied DocNETINF to a set of papers, we faced the problem that the clustering algorithm tended to form very large clusters that were not semantically coherent, i.e., did not represent well-defined concepts. Obviously, the real data is not completely consistent with our model and contains noise. In particular, not all occurrences of technical terms or concepts can be explained by a sparse solution graph, and thus for a fixed graph G found by NETINF, the objective function contains the penalty for many ε-edges. Since concepts are considered to be travelling as one unit, a concept that cannot be explained requires just one ε-edge, independently of its size. Thus, the clustering algorithm can reduce the number of ε-edges and increase the objective function value by combining memes that cooccur at a document and both cannot be explained by a regular edge. For small values of ε, the penalty for ε-edges dominates the penalty for unlikely mutations, and thus the clustering algorithm decides to join unrelated memes. By increasing the concept size, the cascade size can only be increased (each document that contains at least one concept meme belongs to the cascade). Consequently, larger concepts are more likely to have several common ε-edges with other concepts, which makes concepts grow continuously.

[3] http://www.kilgarriff.co.uk/bnc-readme.html

The obvious remedy is the adaptation of the ε-parameter. For our experiments on real data, we increased it from 1e-10 to 0.005. However, this change alone did not solve our problem. Large mutations in large concepts were still not penalized enough to avoid the formation of very large clusters. Consequently, for the application on real data, we changed the computation of the mutation probability from a realization r to a realization r' of a concept γ from the one given in Equation 4.1 on page 22 to

$$P_c(d, d') \propto J(r, r') = \frac{|r \cap r'|}{|r \cup r'|} \tag{7.3}$$

where J denotes the Jaccard similarity. We normalize by all possible realizations $r' \subseteq \gamma$ with $J(r, r') > 0$. The normalization term N is thus defined as

$$N = \sum_{i=1}^{|r|} \sum_{j=i}^{|\gamma|-(|r|-i)} \frac{i}{|r| + j - i} \binom{|r|}{i} \binom{|\gamma| - |r|}{j - i} \tag{7.4}$$

Note that the probability for mutation between non-intersecting realizations is 0. Furthermore, since the number of possible realizations r' of a concept γ grows exponentially with its size, individual mutations become unlikely with growing concept sizes very quickly.

With this definition of the mutation probability, DocNETINF produces intuitive concepts. However, the fact that the algorithm behaves differently on real data than on simulated data and that this change is necessary suggests that there is a problem with the way noise is integrated into the model. In particular, in noisy data, there can be two reasons why the occurrence of a meme at a document cannot be explained (i.e., the incoming edge of this document in the diffusion tree is an ε-edge): Either, it belongs to a concept and the occurrence of the concept can actually be explained (the meme was added in the mutation process), or, it is unexplainable due to noise or because the model is not accurate. In the former case, it is correct to add the corresponding meme to a concept. In the latter case, adding the meme to another concept can reduce the number of ε-edges as described above (overlapping ε-edges only count once per concept), but is semantically wrong. Thus, the gain or loss of a specific cluster of memes is distorted by the way we model noise, and how to avoid this distortion is a question for further research.

7.3 Demo Application

To allow for the exploration of the inferred network and concepts and to give an impression of how a document network with associated concepts could be put to use, we built a demo application. It is a web application that can

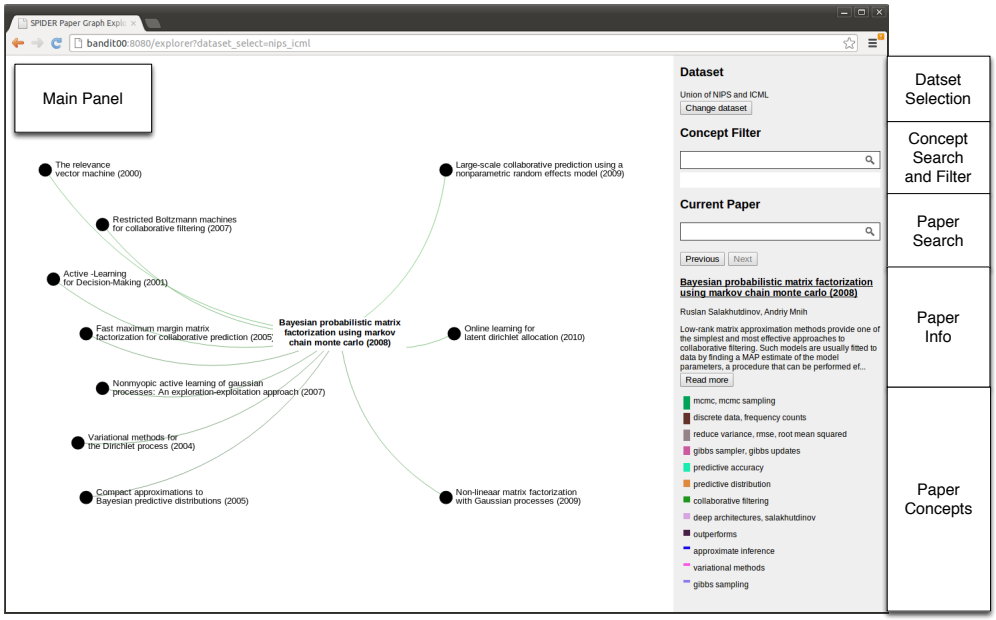


Figure 7.1: Screenshot of the demo application, a web application that allows for the navigation of the document network and associated concepts. The main areas are annotated.

load and display precomputed document networks and concepts. A screenshot of the application with annotated areas is shown in Figure 7.1. The *main panel* shows the currently active paper, its parents in the directed acyclic graph on the left, and its children on the right. The application is bound to the CiteSeerX MySQL database from which it retrieves paper metadata (authors, abstract) to give the user a quick first impression of a considered paper. Furthermore, the application shows the main concepts extracted from the paper. These are selected based on the frequency of occurrence in the paper and the overall likelihood of the concept and corresponding cascade.

The user can make a neighboring paper the currently active one by clicking on it. Alternatively, she can search for a paper or author to directly jump to a paper. To find out more about the connection between the active papers and one of its neighbors, the user can hover the neighboring paper. This makes a tooltip appear that lists the concepts which are inferred to be diffused from/to the neighboring paper. Furthermore, if the user would only like to see the edges that are associated with specific concepts, she can click on a concept in the concept box to add it to the *concept filter*. When the concept filter is active, only neighbors for the filtered concepts are shown.

Data	Extracted	Disambiguated
Venue	589,365 (28% of papers)	not disambiguated
Year	1,096,882 (51% of papers)	-
Author mentions	6,052,516	2,701,295 (45%)
Citations	41,505,622	8,493,645 (20%)

Table 7.1: Overview of the availability of the most important metadata in the CiteSeerX database. *Extracted* denotes the number of entries that are available as simple data type, and *Disambiguated* denotes the number of entries that refer to identified entities.

7.4 Results

In this section, we show results of executing DocNETINF on real data. We briefly describe the CiteSeerX dataset we used for our experiments, discuss quantitative results and give a qualitative anecdotal analysis. Finally, we discuss the quality of the results and remaining problems.

7.4.1 The CiteSeerX Dataset

CiteSeerX[4] is a platform that collects, stores, and makes available scientific papers mainly from the Computer Science area. It allows a user to search and navigate documents similarly to e.g., Google Scholar[5]. The dataset we use for our experiments on real data are subsets of the papers collected by CiteSeerX.

In particular, we worked with a dump of their MySQL database that has a size of 42 GB and contains data about 2,132,204 papers (1,794,726 of them are detected as unique). Beyond simple metadata like authors, publication venue, and publication year, the database also contains citation links between papers. Furthermore, each paper's plain text is available as a simple text file. Since the metadata is extracted automatically from the paper texts, it is neither complete nor completely accurate if available. Table 7.1 shows an overview of the available metadata.

We show results on the *NIPS* (1695 papers) and *NIPS/ICML* (2935 papers) subsets, each containing the papers published at the corresponding conferences. Note that we do the subset selection based on the venue name extracted from the paper, so we do not extract papers whose venue name is missing or incorrect. Furthermore, our algorithm requires document timestamps, so we only select papers whose database entry contains the publication year. We remove duplicates based on a similarity threshold.

[4] citeseerx.ist.psu.edu/
[5] scholar.google.com

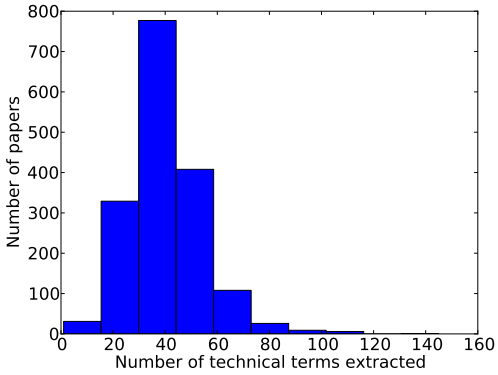
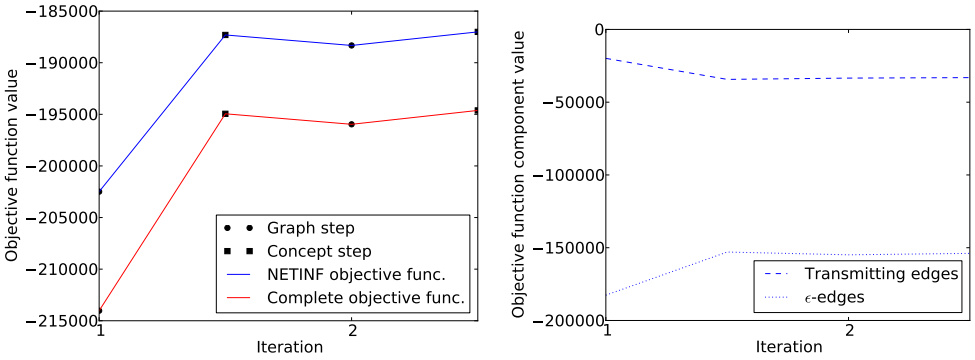


Figure 7.2: Histogram of the number of technical terms extracted for the papers in the NIPS dataset with parameters $\tau = 5$, $df_{min} = 5$, and $df_{max} = 100$.



(a) Objective function.

(b) Objective function components.

Figure 7.3: Objective function development on the NIPS dataset.

7.4.2 Quantitative Results

For the NIPS dataset, we chose the parameters $\tau = 5$, $df_{min} = 5$, and $df_{max} = 100$ which led to a meme vocabulary of size 5098. On average, a paper contains 40 memes. A histogram of the number of memes per paper is shown in Figure 7.2.

We run DocNETINF with the parameters $\beta = 0.5$, $\varepsilon = 0.005$, $\eta = 0.15$, and $\sigma^2 = 3$. The number of concepts d was set to 2000. Figure 7.3 shows the behavior of DocNETINF on the dataset. The objective function flattens out more quickly than on the simulated data. Furthermore, it shows a more erratic behavior. In particular, it decreases in the graph step of the second iteration. Note that this is possible since NETINF is not guaranteed to find the maximum of the objective function. The graph changes by 50.5% from

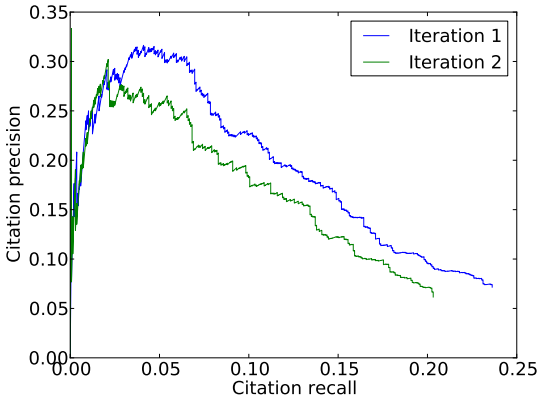


Figure 7.4: Agreement of the inferred graph with the citation graph on the NIPS dataset.

the first to the second iteration, and the concepts change by 65.3%. In Figure 7.3b, we show the gain for transmitting and ε-edges (the corresponding components of the NETINF objective function). In contrast to the behavior on simulated data (cf. Figure 6.3b on page 35), the transmitting edge likelihood does not increase. This can be attributed to the fact that mutations are penalized more with the Jaccard similarity based $P_c(u, v)$ term than with the original one. Furthermore, while the penalty for ε-edges vanishes on simulated data, it dominates the objective function on real data. As discussed in Section 7.2, ε-edges can give an incentive to cluster memes with only little relation. We believe this is the reason why the change of the $P_c(u, v)$ term was necessary.

In Figure 7.4, we show a precision-recall plot of the inferred graph with respect to the available part of the citation graph (we have 1747 citations between the documents). Note that our goal is not to predict citation links. However, the plot shows that there is a correlation of the inferred graph with the citation graph. The fact that the agreement with the citation graph becomes slightly worse from the first iteration to the second reveals a weakness in our model. Since we consider concepts as units, larger concepts do not count more towards the objective function value than smaller ones. Thus the concept formation distorts the similarity of two documents in terms of the set of technical terms used. Adjusting the model to avoid this distortion is left for future research.

7.4.3 Anecdotal Analysis

To give the reader a flavor of our method's results and how they could be used to explore a collection of scientific documents, we present an exem-

plary anecdotal analysis in this section. Note that at the time this analysis was made, we had an additional term in the objective function that penalized large cascades. We only discovered later that this term was actually redundant and removed it. Qualitatively, the results were not affected by this change.

We look at the NIPS/ICML subset of the CiteSeerX dataset (2935 papers). Within this dataset, we focus on a recent paper from the topic modeling domain: *Online Learning for Latent Dirichlet Allocation* by A. Hofmann et al. [13], introduces an online algorithm to infer the parameters of the latent dirichlet allocation topic model for large datasets.

Technical terms In Table 7.2, we show the technical terms extracted from the paper, sorted by term frequency. The n-grams at the top of the list seem to be good keywords. The general theme is represented by *topic models* and *latent dirichlet allocation*. *Online* and *batch algorithm* represent the proposed and the traditional approach, respectively. *Variational bayes*, *variational inference*, and *variational parameters* are keywords for the applied methodology. Many of the extracted n-grams correspond to real technical terms. Since we do not use a linguistic filter, however, some are also verbs, attributes, or frequent combinations of words that are not considered technical terms as such (like *held-out* or *guaranteed to converge*). However, these still represent technical terminology in the sense that they correspond to technical ideas.

Concepts The non-trivial concepts, i.e., the concepts that consist of more than one meme, that are output by DocNETINF for the *Online Learning for Latent Dirichlet Allocation* paper are listed in Table 7.3, sorted according to the average frequency of occurrence of the memes in the paper. Generally, the given concepts seem to cluster semantically related memes. Often, they contain general and more specific versions of the same domain concept (*online - online algorithm*, *mcmc - mcmc sampling*, *corpora - training corpus*, *matrix factorization - non-negative matrix factorization*), almost synonymous memes (*sampling methods - sampling scheme - sampling strategy*), pairs of opposed memes (*non-terminal node - semantic structure - terminal node*), or various aspects of a concept similar to a topic in topic modeling (*blei - latent dirichlet allocation - latent topics - number of topics - topic assignments - topic distributions - topic models - ...*). A few of the clusters are difficult to interpret and might arise from wrong decisions in the clustering process.

Network In Figure 7.5, we show a small subgraph of the graph that DocNETINF inferred for the NIPS/ICML dataset. The node set is selected by an undirected breadth first search from the *Online Learning for Latent Dirichlet Allocation* paper that is stopped after 20 nodes are reached. All inferred

online	factorized distribution
topic models	find a solution
topic assignments	generative process
mcmc	gradient computation
batch algorithm	gradient methods
corpora	gradient optimization
stationary point	guarantee convergence
learning parameters	held-out data
held-out	hierarchical model
large datasets	independent samples
latent dirichlet allocation	inference methods
number of documents	kullback-leibler divergence
inference algorithm	large collection
variational bayes	large scale
variational inference	least-squares
variational parameters	locally optimal
word counts	local optimum
algorithm converges	multinomial distribution
bayesian model	newton method
guaranteed to converge	number of observations
hierarchical bayesian model	number of topics
approximate inference	number of unique
batch size	online learning
coordinate ascent	optimization algorithm
first-order	optimality equation
fisher information matrix	particle filtering
gradient algorithm	point estimate
hessian	positive definite matrix
latent structure	predictive performance
matrix factorization	previous studies
mini-batch	randomized algorithm
natural gradient	resampling
online algorithm	sampling methods
optimization approach	semantic structure
training corpus	show convergence
variational distribution	squared loss
approximate inference algorithms	stationary distribution
bayes estimation	stochastic gradient
bayesian inference	topic distributions
blei	topic proportions
density function	true posterior
empirical bayes	variational bayesian
equivalent to minimizing	variational bound
expectation-maximization	weighted average

Table 7.2: Extracted technical terms for the paper *Online Learning for Latent Dirichlet Allocation*, sorted by term frequency in the paper.

online, online algorithm
mcmc, mcmc sampling
batch algorithm, incremental learning
large datasets, scalability
corpora, training corpus
blei, latent dirichlet allocation, latent topics, number of topics, ...
...topic assignments, topic distributions, topic models, topic proportions, ...
...unique words, word distributions
variational approximation, variational parameters
hierarchical bayesian, hierarchical bayesian model
held-out, held-out data
variational bayes, variational bayesian
deconvolution, natural gradient, source signals
coordinate ascent, model estimation
mini-batch, rbms
matrix factorization, non-negative matrix factorization
method for solving, optimization approach
active learning methods, batch size
fisher information, fisher information matrix
first-order, first-order logic
latent representation, latent structure
full hessian, gradient optimization
computational aspects, deterministic algorithm, randomized algorithm,...
...training distribution
sampling methods, sampling scheme, sampling strategy
large collection, stop words
class of learning, show convergence
directed graphical model, directed models, factorized distribution
non-terminal node, semantic structure, terminal nodes
general representation, image dataset, variational bound
stochastic gradient, stochastic gradient descent
online learning, online learning algorithm
number of particles, observation model, particle filtering
bayes estimation, empirical analysis, future predictions, lars,...
...selection step, stopping point
left hand, previous studies
bayes risk, empirical bayes, variational em algorithm
positive definite matrix, symmetric positive
conjugate gradient methods, gradient methods
approximating distribution, true posterior
least-squares, least-squares estimate
empirical loss, squared loss
head word, mcnemar test, number of unique, regularization penalty

Table 7.3: Concepts of size greater 1 for *Online Learning for Latent Dirichlet Allocation*, sorted by average frequency in the paper.

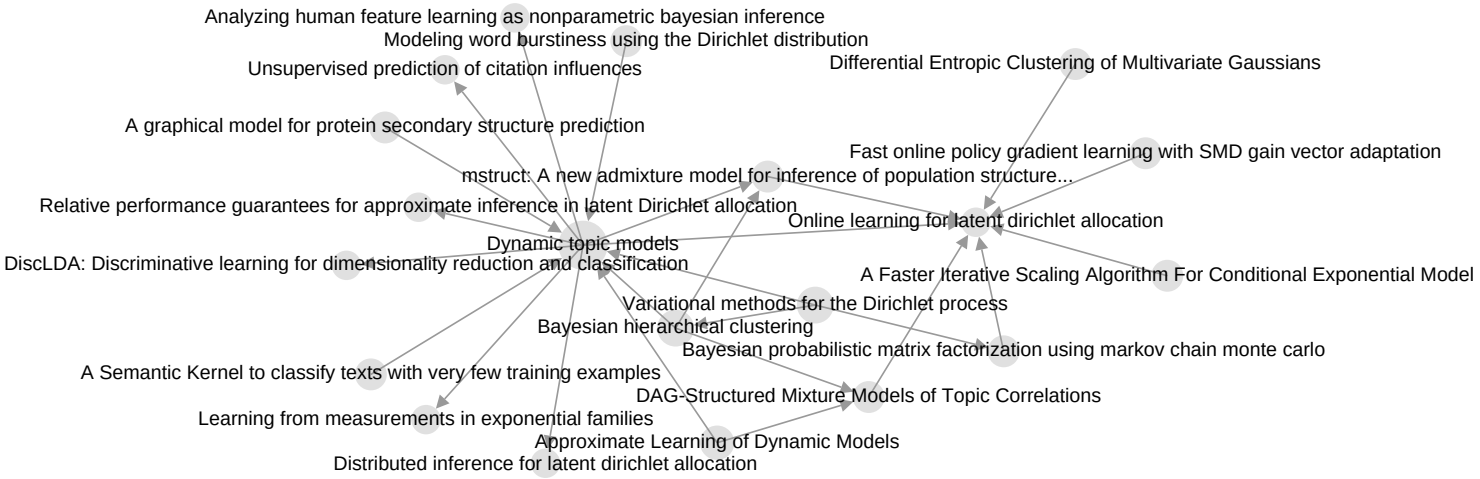


Figure 7.5: A small subgraph for the NIPS/ICML dataset.

mcmc, mcmc sampling
approximating distribution, true posterior
mini-batch, rbms
matrix-factorization, non-negative matrix-factorization
directed graphical model, directed models, factorized distribution

Table 7.4: Selected concepts (5 out of total 11) that are inferred to have travelled from *Bayesian probabilistic matrix factorization using markov chain monte carlo* to *Online Learning for Latent Dirichlet Allocation*.

edges between the displayed papers are shown. The nodes are scaled according to their out-degree. Note that for reasons of readability, the layout does not reflect time.

The paper *Online Learning for Latent Dirichlet Allocation* has no outgoing edges since it is relatively recent (publication year 2010). Incoming edges include *Bayesian probabilistic matrix factorization using markov chain monte carlo* by R. Salakhutdinov et al. [24]. Exploration of the edges with the demo application reveals that this paper is the considered paper's parent in the diffusion trees for the concepts given in Table 7.4. It employs *Markov chain monte carlo* (mcmc) methods, while the considered paper reviews them, and conversely, [24] reviews variational inference methods (*approximating distribution - true posterior*), while the considered paper applies them. The main commonality of the papers is that they both deal with graphical models (*directed graphical model - directed models - factorized distribution*) that can also be interpreted as probabilistic matrix factorizations (*matrix-factorization, non-negative matrix-factorization*). Furthermore, both papers use mini-batches for learning the models (*mini-batch - rbms*). Thus, [24] might contain interesting

variational approximation, variational parameters
coordinate ascent, model estimation
non-terminal node, semantic structure, terminal nodes
approximate inference algorithms
hierarchical model
word counts
latent representation, latent structure

Table 7.5: Concepts that are inferred to have travelled from *Topic Models* to *Online Learning for Latent Dirichlet Allocation*.

further information for a user interested in the considered paper, and the concepts that are associated with the edge between the papers can give her a quick hint about the commonalities.

Another detected parent of the considered paper is *Dynamic Topic Models* [6] by the same main author D. Blei et al. It turns out, however, that the paper with this name in the CiteSeerX dataset is not *Dynamic Topic Models*, but a general topic modeling review paper *Topic Models* by D. Blei et al. [5]. It has been inferred to influence our considered paper with the typical topic modeling concepts (cf. Table 7.5). For a user who wants to acquire background knowledge necessary to understand the considered paper, it would be useful to consult this paper.

Note that the original *Latent Dirichlet Allocation* paper by D. Blei et al. [7] which one would expect to appear as a parent of the considered paper is missing. It was published in the Journal of Machine Learning Research and is thus not part of our data subset.

7.4.4 Discussion

Due to time constraints, we were not able to conduct a user-study to test the results of our algorithm. When we tested the demo application, however, we found that it is useful to explore a paper collection. Furthermore, we prepared datasets of interest (papers published at the NIPS and ICML conferences, as well as papers published at the VLDB, SIGMOD, and ICDE conferences) for two professors of our institution, and gave them access to the tool. They searched for well-known papers, papers related to their research, or their own papers, and assessed the quality of the concepts and connections. Feedback we received included “This tool is fun.”, “The technical terms make a lot of sense.”, “Some inferred neighbors of a paper are good, some are less closely related.” Some concepts were hard to comprehend, and several connections were inferred to explain technical terms or concepts that are not necessarily key concepts in both papers. In systems-related papers, for example, terminology related to the conduction of exper-

iment led to connections between papers, the only semantic similarity being that they both conducted experiments. This suggests that the question of how to detect and use the importance of a concept in a paper is critical for further research. An intuitive idea would be to incorporate the frequency of occurrence of a technical term into the model. Another issue we observed is that often the papers inferred as parents or children are redundant in the sense that they do explain different groups of technical terms, but these are only minor variations of the actual domain concepts. Since our concepts are so fine-grained, they are sensible to slight variations of the exact terminology used, and thus not always able to cluster all technical terms that really belong to a common idea or domain concept.

Chapter 8

Conclusion

Motivated by the information overload in the scientific community, we addressed the task of inferring a network of diffusion and influence between scientific papers. Although scientific papers are usually linked by citations, citations are not always available in an extracted form. Furthermore, they are generally not annotated, i.e., it is difficult to quickly grasp the relationship between a citing and a cited paper. Finally, our approach could potentially be extended to domains where citations are not as ubiquitous as in the scientific domain (e.g., news, non-fiction books, or law). Our goal was to infer a network of papers that represents semantic similarities for the sake of exploration rather than the prediction of actual citation links.

To this end, we took an epidemiological view of information diffusion: The memes - keywords, technical terms or short phrases of technical terminology - that represent ideas and constitute the content of a document are contagions. They infect documents and diffuse over an underlying graph of influence between them. We modeled this diffusion process according to the independent cascade model on a directed acyclic graph of timestamped documents, and showed that this model is a special case of the more general model of the diffusion of contagions in a social network. The NETINF algorithm [12] recovers such a network given a set of observed cascades, and can thus also be applied to infer a network of documents based on the memes appearing in the documents.

To make the model more realistic and the results of the algorithm more useful, we developed a more complex diffusion model based on unobserved concepts - sets of related memes that represent aspects or variations of a domain concept. We defined a generative model for the creation and diffusion of concepts over a document network, where the concepts' realizations (subsets of concept memes used in a particular document) mutate in the diffusion process. Based on this model, we derived an objective function and proposed the DocNETINF algorithm to heuristically maximize this objective

function. It alternates between a network inference and a meme clustering step to jointly infer the unobserved graph and concepts. An evaluation on simulated data showed that DocNETINF recovers the document graph and the concepts clearly better than baseline methods.

Finally, we applied DocNETINF to paper collections from the CiteSeerX dataset. We discussed the problems that arose on the real data and our solution attempts. The achieved results are semantically meaningful, which we demonstrated by anecdotal discussion. Without a proper tool, however, making use of the results is difficult. Thus, we built a web application that allows a user to navigate a preprocessed collection of scientific papers, inspect the inferred concepts, and explore the connections between papers. The feedback we received from two Professors of our institution was generally positive.

Limitations of the presented work are the following: On the one hand, our model considers documents as sets of memes or concepts, respectively. We ignore information about the frequency of occurrence which could potentially be used to make the model more realistic and improve the quality of the results on real data. Furthermore, we assume concepts to be non-overlapping which is a simplification of reality and limits the expressiveness of the model. Future research could address the extension of the model to overcome these two limitations. On the other hand, on real document collections, the algorithm showed a tendency to find large concepts since this can avoid penalty for non-existent edges. By minor adaptations, we could tune the algorithm to produce semantically reasonable results. However, we suspect that the way we model noise can give the clustering algorithm wrong incentives to cluster memes and form large concepts. Investigating alternative ways of including noise into the model is an interesting question for future research.

The results of this thesis suggest that it is possible to jointly infer a diffusion graph and the exact form of the contagions, which are concepts in our case. In particular, estimates of the graph and the concepts can be improved in an alternating way. Furthermore, document networks that are inferred based on a model of information diffusion can capture semantic relationships between documents.

Appendix A

Appendix

A.1 Table of Symbols

Symbol	Description
$G(V, E)$	Document graph
$d \in V$	Document from node set V
t_d	Timestamp of document d
$m \in M$	Meme from meme vocabulary M
$\gamma \in \Gamma$	Concept: Set of memes $\gamma \subseteq M$
$r_{\gamma,d}$	Realization of concept γ in document d, $r_{\gamma,d} \subseteq \gamma$
p_{init}	Probability to add a meme from a concept to the initial realization
p_{add}, p_{del}	Probability to add/delete a meme from one realization to the next
$c \in C$	Cascade: documents that have adopted a meme m or concept γ
R	Realizations of a concept γ in documents
M_d	Memes adopted by document d (document content)
$T(V_T, E_T)$	Diffusion tree for a cascade c
$\mathcal{T}_c(G)$	Set of possible diffusion trees for cascade c in G

Bibliography

[1] Web of knowledge. http://wokinfo.com/. Accessed: 07/03/2013.

[2] Eytan Adar and Lada A Adamic. Tracking information epidemics in blogspace. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 207–214. IEEE, 2005.

[3] Norman TJ Bailey et al. *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.

[4] D. Blei, L. Carin, and D. Dunson. Probabilistic topic models. *Signal Processing Magazine, IEEE*, 27(6):55–65, Nov.

[5] D. Blei and J. Lafferty. Topic models. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.186.4283. Accessed: 20/03/2013.

[6] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[8] Jonathan Chang and David Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88, 2009.

[9] Khalid El-Arini and Carlos Guestrin. Beyond keyword search: Discovering relevant scientific literature. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 439–447. ACM, 2011.

[10] Niklas Elmqvist and Philippas Tsigas. Citewiz: a tool for the visualization of scientific citation networks. *Information Visualization*, 6(3):215–232, 2007.

[11] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. Automatic recognition of multi-word terms:. the c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130, 2000.

[12] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):21, 2012.

[13] Matthew D Hoffman, David M Blei, and Francis Bach. Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 23:856–864, 2010.

[14] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[15] Ioannis Korkontzelos, Ioannis Klapaftis, and Suresh Manandhar. Reviewing and evaluating automatic term recognition techniques. *Advances in Natural Language Processing*, pages 248–259, 2008.

[16] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D Sivakumar, Andrew Tomkins, and Eli Upfal. Stochastic models for the web graph. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 57–65. IEEE, 2000.

[17] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 641–650, New York, NY, USA, 2010. ACM.

[18] Jure Leskovec, Ajit Singh, and Jon Kleinberg. Patterns of influence in a recommendation network. *Advances in Knowledge Discovery and Data Mining*, pages 380–389, 2006.

[19] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[20] Cindy Xide Lin, Qiaozhu Mei, Jiawei Han, Yunliang Jiang, and Marina Danilevsky. The joint inference of topic diffusion and evolution in social communities. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 378–387. IEEE, 2011.

[21] Seth A. Myers and Jure Leskovec. On the convexity of latent social network inference. *CoRR*, abs/1010.5504, 2010.

[22] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[23] Everett M Rogers. *Diffusion of innovations*. Simon and Schuster, 1995.

[24] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using mcmc. *ICML'08*, 2008.

[25] Dafna Shahaf and Carlos Guestrin. Connecting the dots between news articles. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–632. ACM, 2010.

[26] Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Metro maps of science. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 1122–1130, New York, NY, USA, 2012. ACM.

[27] Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Trains of thought: generating information maps. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 899–908, New York, NY, USA, 2012. ACM.

[28] Benyah Shaparenko and Thorsten Joachims. Information genealogy: uncovering the flow of ideas in non-hyperlinked document databases. In *Conference on Knowledge Discovery in Data: Proceedings of the 13 th ACM SIGKDD international conference on Knowledge discovery and data mining*, volume 12, pages 619–628, 2007.

[29] Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *Neural Information Processing Systems*, volume 15, 2003.

[30] Joachim Wermter and Udo Hahn. You can't beat frequency (unless you use linguistic knowledge)-a qualitative evaluation of association measures for collocation and term extraction. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 44, page 785, 2006.

[31] Rongjing Xiang, Jennifer Neville, and Monica Rogati. Modeling relationship strength in online social networks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 981–990, New York, NY, USA, 2010. ACM.