UNIVERSITY OF CALIFORNIA
RIVERSIDE

Algorithmic Challenges in Social Media Search

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Theodoros Lappas

December 2011

Dissertation Committee:
Dr. Vassilis J. Tsotras, Chairperson
Dr. Michalis Faloutsos
Dr. Eamonn Keogh
Dr. Chinya V. Ravishankar

The Dissertation of Theodoros Lappas is approved by:

Dr. Michalis Faloutsos

Dr. Eamonn Keogh

Dr. Chinya V. Ravishankar

Dr. Vassilis J. Tsotras
Committee Chairperson

University of California, Riverside

ACKNOWLEDGMENTS

It is a pleasure to thank the many people who supported me during my graduate studies and made this thesis possible. First and foremost, I would like to deeply thank Dr. Dimitrios Gunopulos, who was my advisor for the first four years of my PhD. I will always cherish his advice and be grateful to him for being an inspiring mentor and teaching me how to be a computer scientist. I would also like to thank Dr. Vassilis Tsotras for being a great advisor during the final year of my studies and for always being there when I needed him. I would like to express my gratitude to Dr. Michalis Vazirgiannis for mentoring me during my undergraduate studies and encouraging me to travel to the USA and pursue a PhD in Computer Science. Further, I would especially like to thank Dr. Evimaria Terzi for being a trusted friend and invaluable collaborator throughout my studies. I would like to give thanks to Dr. Eamonn Keogh and Dr Michalis Faloutsos for being in my committee and for all the wonderful and creative talks we had during the five years of my studies. I would also like to thank Dr. Chinya Ravishankar for accepting my invitation and taking the time to be a member of my committee. I would like to thank my friend and co-author Dr. Michalis Vlachos for giving me the chance to work with him during my stay in Switzerland.

In addition, I would like to thank Amy Ricks for making sure that my paperwork and other responsibilities as a student were always taken care of, allowing me to focus on my research. I would to express my gratitude to my lab-mates for putting up with me and making my days in the lab enjoyable. In alphabetical order: Dr. Benjamin Arai, Dr. Petko Bakalov, Wenyu Huo and Marcos Vieira. I would like to thank my many friends and collaborators who were always there to support me and help me become a better person and computer scientist. In alphabetical order: Muzaffer Akbay, Mustafa Arslan, Astro David Ashtaralnakhai, Dr. Georgos Chatzimilioudis, Dr. Ioannis Drougas, Francesco Fusco, Dr. Aris Gionis, Dr. Tyrone Grandison, Dr. Marios Iliofotou, Dr. Gentian Jakllari, Dr. Vana Kalogeraki, Marios Kokkodis, Dr. Kun Liu, Dr. Heikki Mannila, Dr. Anastasios Mourikis, Dr. Konstantinos Pelechrinis, Dr. Kunal Punera, Dr. Thomas Repantis, Dr. Tamas Sarlos, Petros Venetis, and many more.

Last but not least, I would like to deeply thank my parents, Andreas and Maria, as well as all my friends back in Greece who always provided me with the support and motivation that I needed to complete my studies.

To my parents, Andreas and Maria

ABSTRACT OF THE DISSERTATION

Algorithmic Challenges in Social Media Search

by

Theodoros Lappas

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2011
Dr. Vassilis J. Tsotras, Chairperson

The timeframe of my PhD studies has coincided with the emergence and worldwide spread of social media. These include blogging and microblogging platforms (e.g. Blogger.com and Twitter.com), social networking sites (e.g. Facebook.com and MySpace.com), as well as platforms that allow for the sharing and annotation of content (e.g. Flickr.com and YouTube.com). The popularity and versatility of social platforms has lead to the accumulation of overwhelming volumes of diverse information. As demonstrated by numerous research works, mining such data can further our understanding of these platforms and help us improve the online social experience of their users. My own work has focused on addressing some of the major algorithmic challenges that emerge in the process of mining social data. In particular, I have always found search-based problems to be the most intriguing. On a high-level, the primary objective of my research has been to bridge the gap between users and information in a social context. From a research point of view, I have always been interested in mining two particular types of corpora: graph structures and textual data, both of which are abundant in social media. In this document, I discuss the relevant problems that I have tackled during my studies. The discussion of each problem is accompanied by an appropriate formulation, algorithmic techniques for its solution, and extensive experimental evaluations.

Contents

List of Figures

Chapter 1

Introduction

The emergence and worldwide spread of social media has lead to the accumulation of massive volumes of textual, as well as graph-based data. The diversity of such data is impressive. Social platforms are hosts to multiple types of user-submitted text, available in different forms. Further, in the context of social networking platforms, users have the opportunity to interact with each other in various ways. Such interactions can be intuitively visualized as graph structures, with users representing nodes and edges representing the various types of connections. While each of these two types of data can be a very rich source of information, it also presents its own unique problems and algorithmic challenges. The focus of my own work has been on identifying and tackling the problems that users face in their effort to process overwhelming amounts of diverse textual and graph-based data, and retrieve the information that is relevant to their interests and needs. Next, I briefly introduce the particular problems that my research has focused on. The problems I have tackled in my work can be grouped into two categories, based on whether they are defined in the context of textual domains or graph structures.

1.1 Textual Data

During my studies, I have worked with different types of text that can be found in social media. These include user comments, discussions, tags, reviews, blog-posts, news articles and many more. Even though each type of textual content requires special consideration and mining techniques, the common denominator of my work has always consisted of two principal components: a) a search-based formulation of the problem, aiming at bridging the gap between users and the valuable information that can be mined from textual content and b) the presentation of efficient methods that can handle very large corpora. As I have discovered through my research, each type of text comes with its own unique set of challenges, thus requiring customized approaches. Next, I discuss the different types of textual content that I have worked on, as well as the respective problems that I have tackled for each type.

Bursty Text [1]: The online nature of the World Wide Web ensures the accumulation of large volumes of text, arriving in a streaming fashion from various sources. Consider a long document sequence, formed by blogs or news portals, over a large period of time. The articles in such collections cover newsworthy events that took place at various times, and are of great interest to the user. Therefore, it is crucial to have methods that can mine large volumes of data in an online setting and use the extracted knowledge toward event discovery. We observe that each event is characterized by a set of descriptive keywords, revealing basic information such as the place where the event occurred, or the names of the persons involved. For the duration of the events lifespan and consequent coverage in the news, these characteristic terms appear repeatedly in relevant articles, leading to uncommonly high frequencies (bursts). A major contribution of our work is a formal definition of term burstiness, based on the concept of discrepancy []. Based on our definition, we present a parameter-free, linear-time algorithm to identify the time-intervals that maximize the burstiness score of any given term. We then propose an efficient search framework that utilizes the mined to identify and rank documents in the context of a user-submitted query. This allows users to search for events that match their own interests, as encoded in a query of terms. Our work is the first that directly incorporates burstiness in the indexing and ranking of documents, leading to a complete burstiness-aware search framework.

The temporal burstiness problem assumes a single stream of documents and focuses exclusively on the *temporal* burstiness of terms. In practice, however, we can have multiple text streams originating at different locations. Consider, for example, a set of Twitter feeds from users living in different parts of the world. In such a context, the impact of an event should be measured not only based on how long it stays in the spotlight, but also on its *spatial impact*. For example, a highly influential event of global importance (e.g. the financial crisis) is likely to appear in the news from multiple countries across the world for an extended period of time. On the other hand, a news-worthy, yet less influential event (e.g. a small-scale earthquake) will only be covered in the streams that are close to the event's source, and for a shorter timeframe.

In our work, we initiate a study on formalizing and identifying patterns that are bursty in both time and space. We then explore how these patterns can be used toward the retrieval of documents on events with high spatiotemporal impact. Our goal is to efficiently tackle the following problem: *Given a set of terms q submitted by the user, return the relevant documents with the highest spatiotemporal impact in the collection of the streaming document sources*. To the best of our knowledge, our work is the first to utilize the spatiotemporal burstiness of terms in order to return relevant documents on events that are relevant to the user's query, and also have a significant spatiotemporal impact. We present two complementary approaches for the two problems, providing an informative and insightful analysis of the spatiotemporal burstiness of terms from alternative perspectives.

Opinionated Text[2]: Another important type of text that has emerged with the establishment of social media and Web 2.0 platforms is *opinionated text*. Users use the web as a medium to express their opinions on various topics . One of the most representative instantiation of this form of expression comes in the form of *item reviews*. Given an extensive corpus of reviews on an item, a potential customer goes through the expressed opinions and collects information, in order to form an educated opinion and, ultimately, make a purchase decision. This task is often hindered by false reviews, that fail to capture the true quality of the item's attributes. These reviews may be based on insufficient information or may even be fraudulent, submitted to manipulate the item's reputation. In this work, we formalize the *Confident Search* paradigm for review corpora. We then present a complete search framework which, given a set of item attributes, is able to efficiently search through a large corpus and select a compact set of high-quality reviews that accurately captures the overall *consensus* of the reviewers on the specified attributes. We also introduce **CREST** (Confident REview Search Tool), a user-friendly implementation of our framework and a valuable tool for any person dealing with large review corpora. The efficacy of our framework is demonstrated through a rigorous experimental evaluation.

Multi-Lingual Text: Blog posts, news events and product reviews are nowadays present on the web in multiple languages. How can one deliver to the user a foreign language document that serves both: a) the user's search query (i.e. relevance), and b) the user's understanding of the foreign language (comprehension difficulty)?

Currently, search engines provide search functionality that primarily focuses on the *relevance* of a given query. However, when searching for results other than in one's native language, it is particularly important to consider the user's proficiency level of a given foreign language. In this work, we provide a first step toward this goal, by designing ranking operators for foreign documents based on both their content relevance and easiness of understanding (i.e., comprehensibility). In order to establish the "difficulty" of each foreign document, we deploy foreign language readability measures, based on machine learning and Natural Language Processing (NLP) features of a document. Our evaluations indicate that the comprehensibility level of a document, as provided by our technique, is consistent with the ordering provided by human annotators.

1.2 Graph Structures

Graph structures are at the core of any social system, serving as the means of propagation of a diverse range of content (e.g. textual data, videos, pictures). In addition, the connections of a user can tell us a lot about her preferences and expertise in different areas. Such knowledge can be then used to personalize the social experience of the users or group them based on common characteristics.

A significant portion of my own research has been devoted to addressing interesting problems that emerge in the process of mining such social graphs. Even though my work has not been limited to a single type of graph, the methodology I have followed has been consistent:

1. Identify a graph structure that is prevalent in social platforms.
2. Adopt principled and meaningful definitions for the nodes (i.e. the individuals operating within the system) and edges (i.e. the types of connections between the individuals) of the graph.
3. Formulate an interesting problem in the context of the social network.
4. Design efficient algorithms for the solution of the problem.
5. Provide both quantitative and qualitative experimental evidence that demonstrate the efficacy of the proposed methods.

Driven by this methodology, most of my research endeavors have converged to problems with two primary characteristics: a) a search-based formulation, asking for nodes that serve a particular purpose, and b) a consideration of the complem

Finding a Team of Experts in Social Networks [3]: Given a task T, a pool of individuals $\mathcal{X}$ with different skills, and a social network G that captures the compatibility among these individuals, we study the problem of finding $\mathcal{X}'$, a subset of $\mathcal{X}$, to perform the task. We call this the TEAM FORMATION problem. We require that members of $\mathcal{X}'$ not only meet the skill requirements of the task, but can also work effectively together as a team. We measure effectiveness using the *communication cost* incurred by the subgraph in G that only involves $\mathcal{X}'$. We study two variants of the problem for two different communication-cost functions, and show that both variants are NP-hard. We explore their connections with existing combinatorial problems and give novel algorithms for their solution. To the best of our knowledge, this is the first work to consider the TEAM FORMATION problem in the presence of a social network of individuals. Our experiments show that our framework works well in practice and gives useful and intuitive results.

Searching for Effector Nodes [4] Assume a network (V, E) where a subset of the nodes in V are *active*. We consider the problem of selecting a set of k active nodes that best explain the observed activation state, under a given information-propagation model. We call these nodes *effectors*. We formally define the k-EFFECTORS problem and study its complexity for different types of graphs. We show that for arbitrary graphs the problem is not only NP-hard to solve optimally, but also NP-hard to approximate. We also show that, for some special cases, the problem can be solved optimally in polynomial time using a dynamic-programming algorithm. To the best of our knowledge, this is the first work to consider the k-EFFECTORS problem in networks. We experimentally evaluate our algorithms using the **DBLP** co-authorship graph, where we search for effectors of topics that appear in research papers.

Search-Based Recommendation of Nodes [5] Numerous social networking platforms are giving users the option to endorse *entities* that they find appealing, such as videos, photos, or even other users. We define this model as a *Social Endorsement Network*, visualized as a bipartite graph with edges (endorsements) from users to endorsed entities. In this work, we formalize the problem of *recommendations in social endorsement networks*: given a query of tags and a social endorsement network, the problem is to recommend entities that match the query and also share a significant number of common endorsers. We propose an efficient search engine for the solution of the problem, able to produce high-quality and explainable recommendations. The entire framework is designed in a principled and efficient manner, making it ideal for large-scale systems. Finally, in a thorough experimental evaluation on real datasets, we illustrate the efficacy of our methods and provide some valuable insight on social endorsement networks.

Chapter 2

On Burstiness-Aware Search for Document Sequences

2.1 Introduction

Suppose we are presented with a long *document sequence*, formed by newspaper articles spanning several local titles (e.g., New York Times, The Wall Street Journal, etc.), over a large period of time. Such corpora is becoming increasingly available, due to initiatives such as The National Digital Newspaper Program (NDNP) [6] by the Library of Congress (LC), and other similar ventures for the digitization of periodicals by large corporations such as Microsoft (www.microsoft.com) and Google (www.google.com). The articles in such collections cover newsworthy events that took place at various times. Each event is characterized by a set of descriptive keywords, revealing basic information such as the place where the event occurred, or the names of the persons involved. For the duration of the event's lifespan and consequent coverage in the news, these characteristic terms appear repeatedly in relevant articles, leading to uncommonly high frequencies (bursts). In the typical search paradigm, the user encodes a topic of interest using a query (i.e. a set of keywords), which is then submitted to a search engine. Typical search engines rely on static, frequency-based measures (e.g. *tf-idf*) for the purposes of indexing and querying the underlying collection. These measures record the frequency of a term in each document, typically normalized by a global frequency measure, in order to capture the impact of the term in the entire collection. The underlying assumption is that an occurrence of a term has the same significance, regardless of the moment in time it occurs. Our claim is that, for a contiguous document sequence observed through time, this assumption is invalid: the importance of terms varies through time, as they are used to describe current influential events that are discussed in the corpus. Therefore, it is essential to consider the temporal dimension of the data in the indexing and ranking process. The ultimate purpose of our work is the creation of an efficient, end-to-end framework that, given a document sequence, identifies "bursty" intervals for each term and utilizes this information toward an efficient, burstiness-aware search mechanism.

Even though some work has been devoted to measuring burstiness in different contexts, the concept has yet to be formalized. A major contribution of our work is a formal definition of burstiness that is based on the concept of discrepancy. Discrepancy theory has applications in several fields including machine learning, computer graphics and computational geometry [7, 8, 9, 10]. The concept is generally used to describe the deviation of a situation from the "expected" behavioral baseline. Based on our definition, we present a parameter-free, linear-time algorithm to identify the time-intervals that maximize the burstiness score of any given term. We present the theoretical foundations of our work and proceed to evaluate it thoroughly on a new dataset.

Our Contributions: In this work we make the following contributions:

i. A formal definition of term burstiness in the terms of numerical discrepancy.

ii. A parameter-free, linear-time method to identify the maximum burstiness intervals for a given term.

iii. An efficient search framework for documents, that considers term burstiness in the indexing and ranking process. The framework uses an extension of the well-known TA algorithm [11] for finding the top intervals.

The rest of this work is organized as follows: Section 2.2 discusses related work. Section 2.3 describes the basic notation used in this work. Section 2.4 introduces our definition of burstiness and discusses efficient techniques for the identification of bursty intervals for a given term. In Section 2.5, we present two different versions of a complete, burstiness-aware search framework. Finally, we conclude with a thorough experimental evaluation in Section 2.6.

2.2 Related Work

The concept of burstiness has been studied in several domains. A significant portion of this work has been inspired by Kleinberg's seminal paper on the bursty and hierarchical structure of streams [12]. We discuss Kleinberg's approach in more detail in Section (2.6.2). A considerable amount of work has been devoted to developing efficient burst-detection methods [13, 14, 15, 16]. Even though we propose a method of our own, we do so in the process of creating a complete search framework, which is the main contribution of our work. The main benefits of our method are that it runs in linear-time and is also completely parameter-free. This makes it ideal for very large sequences of documents, spanning significant periods of time. That being said, our search framework is compatible with *any* burst detection method that can report non-overlapping bursty intervals and their respective scores, for any given term.

Another burst-detection method is presented by Fung et al.[13]. In this work, bursty terms are clustered to represent events discussed in the data. In [14], the authors classify terms in four burstiness categories, based on their frequency trajectory. Their use of spectral analysis is similar to the one used by Vlachos et al. in [15], where the authors focus on periodic and bursty artifacts in query logs. In [16], the authors use a wavelet-based structure for aggregate monitoring of data streams.

Burstiness has also been evaluated in the context of other applications, such as stream clustering [17], and even in the context of graphs [18]. Further, He et al. [19] apply Kleinberg's model to topic clustering.

Bansal and Koudas [20, 21] have presented a system for the analysis of streaming blogs. Even though no details on the employed methods are given, their work is relevant to ours, in that they ultimately map bursty terms to specific blogposts. To the best of our knowledge, our work is the first that directly incorporates burstiness information in the indexing and ranking of documents, leading to a complete burstiness-aware search framework.

2.3 Preliminaries

In this work, we explore corpora that are formed as a sequence of documents, spanning a pre-defined timeline. For a timeline of m consecutive timestamps, we define a *document sequence* $\mathcal{S}$ as:

$$\mathcal{S} = S_1, S_2, ..., S_m \tag{2.1}$$

where S_i represents the set of documents appearing on the i_{th} timestamp. Further, we define the *frequency sequence* Y_t for a given term t as:

$$Y_t = y_{t1}, y_{t2}, ..., y_{tm} \tag{2.2}$$

where y_{ti} expresses the frequency of term t on the i_{th} timestamp of the timeline. We assume that y_{ti} is equal to the number of documents in S_i that include term t. Finally, by $Y_t[l : r]$, we represent an interval of Y_t that includes all timestamps from y_{tl} to y_{tr} (inclusive). Note that the words "interval" and "segment" are used interchangeably in our work.

2.4 Defining Term Burstiness

In this section we present a formal definition of term burstiness in the terms of *numerical discrepancy*. We then show how the problem of finding the maximum burstiness intervals can be solved in linear time.

2.4.1 A Discrepancy Model of Burstiness

We first present the general definition of numerical discrepancy [7, 8, 9]. Let $\mathcal{P}$ be a set of points distributed over random locations in $[0, 1]^d$, where d is the number of dimensions on the plane. For any region $\mathcal{R}$ in $[0, 1]^d$, let $\mu(R)$ be the Euclidean measure of $R \cap [0, 1]^d$ (i.e. the area of R), and $\mu_{\mathcal{P}}(R)$ be the discrete measure $|R \cap \mathcal{P}|/|\mathcal{P}|$ (i.e. the fraction of points of $\mathcal{P}$ inside R). Then, the *numerical discrepancy* of R with respect to P is defined as:

$$\mathcal{D}_{\mathcal{P}}(R) = |\mu(R) - \mu_{\mathcal{P}}(R)| \tag{2.3}$$

Even though the concept is meaningful for any $d > 1$, we will focus on the one-dimensional case, suitable for our sequence representation. For $d = 1$, a region R is reduced to a one-dimensional interval I, defined within the unit interval $[0, 1]$. Following Equation (2.3), the discrepancy of a given interval I is defined as the absolute value of the difference between its length and the ratio of points from $\mathcal{P}$ that fall within I:

$$\mathcal{D}_{\mathcal{P}}(I) = |\mu(I) - \mu_{\mathcal{P}}(I)| = \left| len(I) - \frac{|\mathcal{P} \cap I|}{|\mathcal{P}|} \right|, \tag{2.4}$$

where $len(I)$ is the length (i.e. the euclidean measure) of interval I. Conceptually, $\mu(I)$ expresses the baseline, i.e. the fraction of points from $\mathcal{P}$ that is *expected* to fall within I, while $\mu_{\mathcal{P}}$(I) represents the observed fraction. This constitutes an appropriate definition for term burstiness, which is similarly expressed by increased frequency values that diverge from a term's individual baseline. In the mapping of term burstiness to numerical discrepancy, the set of points $\mathcal{P}$ is represented by the total frequency of a term, observed throughout the entire document sequence. Next, we formally define the baseline $\mu(I)$ and the observed fraction $\mu_{\mathcal{P}}(I)$ in the context of term burstiness.

Depending on the nature of the data, $\mu(I)$ can be either pre-defined or based on the underlying distribution. In [10], the authors explore discrepancy in the context of different distributions (Poisson, Binomial). Even though such an approach may work well in some scenarios, the assumption that the entire dataset can be accurately described by a single distribution is not always valid. In addition, the use of a probability distribution introduces parameters that are not always intuitive to tune. Given an interval $Y_t[l:r]$ of the frequency sequence for a term t, we define the baseline as

$$len(Y_t[l:r]) \times Avg(Y_t),$$

i.e. the average frequency observed over all timestamps, multiplied by the length of the interval. To conform with the definition of numerical discrepancy, we then project

$Y_t[l:r]$ on the unit interval $[0,1]$ by dividing the baseline by $\sum_{i=1}^{n} y_{ti}$.

Formally, let I be the projection of $Y_t[l:r]$ on $[0,1]$. Then, we define the baseline $\mu(I)$ for I as:

$$\mu(I) = \frac{len(Y_t[l:r]) \times Avg(Y_t)}{\sum_{i=1}^{m} y_{ti}} \tag{2.5}$$

By replacing the average and solving further, we get:

$$= \frac{len(Y_t[l:r]) \times (\sum_{i=1}^{m} y_{ti})/m}{\sum_{i=1}^{m} y_{ti}}$$

$$= \frac{len(Y_t[l:r])}{m} = len(I) \tag{2.6}$$

Indeed, the baseline is equal to the Euclidean measure (length) of I, as mandated by

Eq. (2.4).

Next, we define $\mu_{\mathcal{P}}(I)$, the fraction of the term's frequency observed within the interval, as the frequency of t observed within interval $Y[l : r]$, divided by the total frequency of t throughout the sequence:

$$\mu_{\mathcal{P}}(I) = \frac{\sum_{i=l}^{r} y_{ti}}{\sum_{j=1}^{m} y_{tj}} \tag{2.7}$$

By replacing Equations (2.6) and (2.7) in Eq. (2.4), we get:

$$\mathcal{D}_{\mathcal{P}}(I) = \left| \frac{len(Y_t[l : r])}{m} - \frac{\sum_{i=l}^{r} y_{ti}}{\sum_{j=1}^{m} y_{tj}} \right| \tag{2.8}$$

Note that Eq. (2.8) takes positives values if the observation is either *greater* or *less* than the baseline. Conceptually, the latter occurs if the frequency of a term within an interval is *less* than expected. Even though this typically constitutes a case of discrepancy, it is of little value for the purposes of measuring term burstiness. Instead, we would like burstiness to be positive only for **uncommonly high** frequency observations. Thus, given a term t and an interval $[l : r]$ on the timeline, we define the **Burstiness of** t **in** $[l : r]$ as:

$$\mathcal{B}(t, [l : r]) = \left(\frac{\sum_{i=l}^{r} y_{ti}}{\sum_{j=1}^{m} y_{tj}} - \frac{len(Y_t[l : r])}{m} \right) \tag{2.9}$$

2.4.2 Maximizing Burstiness

Using Eq. (2.9), we can measure the burstiness of a given term for any interval on the timeline. The next step is to identify high-burstiness intervals for each term. On a higher level, the problem definition is the following:

Problem 1. Bursty Intervals Problem: *Given the frequency sequence Y_t of a given term t, identify the set of intervals that maximize the Burstiness function $\mathcal{B}(t, \cdot)$.*

Next, we argue that Problem 1 is equivalent to the well-known *maximum sum segments* *problem*, defined as such:

Problem 2. ***Maximum Sum Segments Problem:***

Given an input sequence $X = x_1, x_2, ..., x_n$ of real numbers, identify the K segments with the highest total scores, where the score $f(X[i:j])$ of a segment $X[i:j] = x_i, x_{i+1}, ..., x_j$ is equal to the sum of its elements:

$$f(X[i:j]) = \sum_{k=i}^{j} x_k \tag{2.10}$$

The **Maximum Sum Segments** Problem comes up in different domains and has been extensively researched in the past [22]. To show that Problems (1) and (2) are equivalent, it is sufficient to show that, given a term t, the Burstiness score of any given interval is equal to the sum of the Burstiness values observed in the individual timestamps of the interval. Formally:

$$\mathcal{B}(t,[l:r]) = \sum_{k=l}^{r} \mathcal{B}(t,[k:k]) \tag{2.11}$$

Proof.

$$\sum_{k=l}^{r} \mathcal{B}(t,[k:k]) = \sum_{k=l}^{r} \left(\frac{y_{tk}}{\sum_{j=1}^{m} y_{tj}} - \frac{1}{m} \right)$$

$$= \left(\frac{\sum_{i=l}^{r} y_{ti}}{\sum_{j=1}^{m} y_{tj}} - \frac{len(Y_t[l:r])}{m} \right) = \mathcal{B}(t,[l:r])$$

□

Problem (1) is now reduced to solving the **Maximum Sum Segments** Problem on the *burstiness sequence* $\mathcal{B}_t$, defined as such:

$$\mathcal{B}_t(i) = \mathcal{B}(t,[i:i]) = \left(\frac{y_{ti}}{\sum_{j=1}^{m} y_{tj}} - \frac{1}{m} \right), 1 \leq i \leq m \tag{2.12}$$

Eq. (2.12) assumes the global baseline given by Eq. (2.5). Alternatively, one could use the local average of each interval as a baseline. In that case, consecutive segments of $\mathcal{B}_t$

could be computed separately, and then concatenated to form the entire sequence. This could allow for a more flexible calculation of burstiness, and avoid reporting anticipated periodical bursts (e.g. a burst of the term "Christmas" during December).

The standard formulation of the **Maximum Sum Segments** problem has the following disadvantage: given a high-scoring segment, one can easily generate several others by simply appending or removing a small number of elements. These segments convey little extra information regarding the burstiness of a term. To address this, we adopt a slightly different formulation, based on the concept of the *maximal segment*:

Definition 1. Maximal Segment: *Let X be a non-empty score sequence. A segment $X[i:j]$ is* ***maximal in*** *X if*

i. All proper sub-segments of $X[i:j]$ have a lower score

ii. No proper super-segments of $X[i:j]$ in X satisfies (i).

Figure (2.1) illustrates the Burstiness Sequence $\mathcal{B}_t$ for the term "Earthquake", as it manifested in a daily newspaper over a fixed period of time. The values on the x-axis represent consecutive timestamps. Following Eq. (2.12), negative values correspond to points when the observed frequency was less than the baseline. Here, "WZ" is identified as a maximal segment; conceptually, extending the interval from either side can only reduce its score, since more negative than positive values will be included.

No two maximal sequences can overlap. A formal proof appears in [23], but essentially, given two maximal overlapping sequences, either the union or the intersection of the two would have higher discrepancy than one of the two, creating a contradiction. Thus, every element of the input sequence belongs to exactly one maximal segment. Therefore, for any given sequence of real numbers, there exists a finite set that contains *all* the maximal scoring segments. We can now formalize the problem as such:

Problem 3. ***All Maximal Segments Problem:*** *Given an input sequence $X = x_1, x_2, ..., x_n$ of real numbers, identify the set of all segments of X that satisfy Definition (1).*

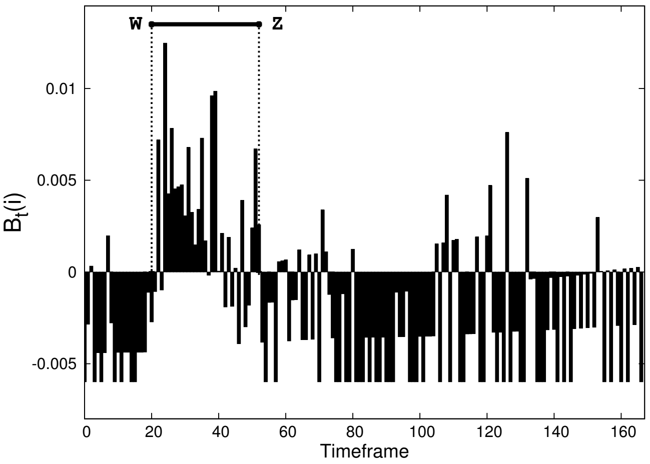


Figure 2.1: Burstiness sequence $\mathcal{B}_t$ for t=“earthquake”

2.4.3 Algorithms for the All Maximal Segments Problem

In [23] the authors present a linear-time algorithm for solving the **All Maximal Segments Problem**. The algorithm accepts as input a sequence of real numbers and reports the set of all maximal segments. For the rest of this work, we refer to this algorithm as MAX-1. The details and pseudocode of the algorithm can be found in [23]. MAX-1 filters out maximal segments with a negative score. This is ideal for the purposes of burstiness evaluation, since negative-scoring intervals represent regions where the observed frequency of a term was less than the expected. Finally, in addition to being linear, the approach is completely parameter-free. Next, we present an extension of MAX-1 and discuss its advantages.

In [12], Kleinberg discusses *anisochronies*, the non-uniform relationships between the time spanned by a story’s events and the amount of time devoted to these events in the actual telling of the story. Considering the coverage of events in news streams (e.g. newspapers, blogs), we identify two primary levels of bursty behavior for the terms describing an event: the first level represents the extended time period when the event was generally discussed in the news. Depending on the nature and significance of the event, this period can be extended to include weeks or even months. The second burstiness level pertains to smaller intervals within this extended period, when the event was particularly popular and extensively covered in the news. In the context of a newspaper, such intervals may represent the first time an event made the headlines, or a new development in an older event that brings it back to the front page.

Conceptually, the intervals reported by `MAX-1` capture the first level of burstiness activity for a given term. By re-applying the algorithm on each of the reported maximal intervals independently, we can easily identify the second-level burstiness intervals. For the rest of this work, we refer to this algorithm as `MAX-2`. The pseudocode is shown in Algorithm (1). Multiple iterations of `MAX-2` could be used to obtain a hierarchical structure of the bursty intervals. As we demonstrate in the Experiments section, a single iteration is enough to capture the burstiness patterns of events.

Algorithm 1 : `MAX-2`

```
Input: $\mathcal{I}$: Set of first-level maximal intervals for $Y_t$
Output: $\mathcal{I}'$: Set of second-level maximal intervals for $Y_t$
$\mathcal{I}' \leftarrow \emptyset$
for every interval $I \in \mathcal{I}$ do
    $\mathcal{I}' \leftarrow \mathcal{I}' \cup$ MAX-1$(I)$ // MAX-1 returns 1st level intervals
Return $\mathcal{I}'$
```

2.5 Query Evaluation

In this section, we describe two different ways to utilize burstiness information to create a complete, burstiness-aware search framework. The described search frameworks constitute the main contribution of our work. Our first approach focuses on indexing and ranking documents directly, while the second approach is more advanced and performs a more informative, interval-based evaluation of a given query. For each approach, we start by discussing the underlying indexing mechanism, and then proceed to discuss the respective query evaluation algorithms.

It is important to note that both approaches are compatible with *any* method than can evaluate the frequency sequence of a term over a specified timeline, and report **non-**

overlapping bursty intervals and their respective scores.

2.5.1 Evaluating Documents

Next, we describe a burstiness-aware query evaluation framework that retrieves and ranks documents based on a given query. First, we discuss the employed indexing mechanism.

Indexing: In the standard inverted index structure, each term is mapped to the list of documents that contain the term. In a more advanced scenario, the document lists are sorted on a pre-computed score that expresses the strength of the connection between the term and the document. In order to use such a structure in our framework, we need to define a formula that evaluates a document with respect to a given term, in the context of burstiness:

Definition 2. *Given a term t and a document d, let $I_{t,d}$ be the bursty interval of t that includes (the timestamp of) d. Then, the* ***Burstiness of*** d ***with respect to*** t *is defined as:*

$$d\text{-}score(t,d) = \left\{ \begin{array}{c} \mathcal{B}(t, I_{t,d}) \times freq(t,d) \ \ , \textit{ if } I_{t,d} \neq \emptyset \\ \\ 0, \ \ \textit{otherwise} \end{array} \right\} \tag{2.13}$$

where $d\text{-}score$ stands for *document score*. Conceptually, $\mathcal{B}(t, I_{t,d})$ returns the burstiness score of $I_{t,d}$, as defined by Eq. (2.9). Also, $freq(t,d)$ returns the frequency of term t with respect to document d. In our experiments, we assume $freq(t,d) = \log(TF(t,d)+1)$, where $TF(t,d)$ returns the number of occurrences of t in d. The logarithm is used to moderate the effect of the frequency and ensure that burstiness is the dominant factor. Finally, if $d\text{-}score(t,d) = 0$, d is not included in the sorted list. Note that

We can now build an inverted index structure, where each term is mapped to a list of documents, sorted on their $d\text{-}score$. Next, we discuss how we can use this index to evaluate multi-term queries.

First, we formally define the *Document Evaluation Problem* as such:

Problem 4. Document Evaluation Problem*: Given a query of terms $q = \{t_0, t_1, ...\}$, retrieve the k documents with the k highest values for $\sum_{t \in q} d\text{-}score(t, d)$.*

Algorithm 2 TA Algorithm

```
Input: query q = {t_0, t_1, ...}, int k
Output: Set of top-k documents
TopK ← ∅  // sorted, holds at most k elements
Threshold T ← 0
$\mathcal{L} = \{L_0, L_1, ..\}$ // set of Doc Lists for each term in q
while (Not All lists in $\mathcal{L}$ Have Been Exhausted) do
    for (every List L ∈ $\mathcal{L}$) do
        cand ← getNext(L)
        total ← cand.score  // Holds cumulative score
        T ← (T − lastSeen(L) + cand.score)
        for (every List L′ ∈ $\mathcal{L}$, L′ ≠ L) do
            total+ = getDScore(L′, cand)
        TopK.insert(cand, total)
        if ((TopK.size() == k) && (TopK.last() >= T))
            return TopK // Early Termination
return TopK
```

- getNext(L) returns the next candidate to be evaluated from list L, under sorted access.
- getDScore(L′, cand) is a random access probe that retrieves the $d\text{-}score$ of the candidate document from list L'.
- lastSeen(L) returns the score of the last candidate seen under sorted access in List L
- TopK.last() returns the score of the lowest-scoring element in the Result.

With an appropriate index structure at our disposal, the next step is to find a query evaluation algorithm to address Problem 4. For this purpose, we use the Threshold Algorithm (TA)[11], an efficient top-k evaluation algorithm, which is able to deal with multi-predicate queries. The algorithm goes through the sorted lists mapped to the terms of a query, evaluating documents in descending order. For every document seen under sorted access in some list, a random access probe retrieves the respective scores of the document from the other lists. The cumulative score is then calculated, and the document is considered as a top-k candidate. The algorithm maintains a threshold value T, based on the score of the last document seen from every List. As soon as k documents with a cumulative score of at least T have been found, the algorithm terminates. The authors prove that, while this mechanism allows for early termination, it does not affect the optimality of the result. Algorithm (2) contains the pseudocode of the TA algorithm. The algorithm is designed so that candidates from different sorted lists can be also evaluated in parallel. In that case, lines 5-11 of the algorithm can be handled by independent threads.

The proposed Inverted Index structure and the TA algorithm compose a complete search framework that efficiently solves the **Document Evaluation Problem**. The framework is thoroughly evaluated in the Experiments Section.

2.5.2 Evaluating Intervals

The framework described in the previous section focuses on the indexing and ranking of documents. In this section, we describe an alternative approach that places the focus on intervals. Given a query of terms, we would like to find periods of time when all terms simultaneously displayed bursty behavior, indicating the occurrence of an underlying event. Next, we describe a search framework for this problem.

Indexing: First, we define a formula that evaluates the burstiness of interval with respect to a given term:

Definition 3. *Let $\mathcal{I}_t$ be the set of bursty intervals for a term t. Then, given a query of terms $q = \{t_0, t_1, ..\}$, an interval I is identified as **bursty with respect to** q, if $\forall t \in q,\ \exists\ I' \in \mathcal{I}_t,\ s.t.\ I \subseteq I'$. Then, the burstiness score of I with respect to q is defined as:*

$$i\text{-}score(I, q) = \sum_{t \in q} B(t, super(\mathcal{I}_t, I)) \ \ , \tag{2.14}$$

*where $i\text{-}score$ stands for interval-score. Also, $super(\mathcal{I}_t, I)$ returns the interval $I' \in \mathcal{I}_t,\ s.t.\ I \subseteq I'$ (i.e. I' is a **super-segment** of I).*

Conceptually, I is bursty with respect to a query if it has been included in a bursty interval for *all* the terms in the query. The $i\text{-}score$ of I is then the sum of the scores of all the bursty intervals that include it. Note that this definition requires the bursty-interval set $\mathcal{I}_t$ for a given term t to consist of **non-overlapping intervals**. This guarantees that **at most one** interval $I' \in \mathcal{I}_t$ is a super-segment of I. Using Eq. (2.14), we can now build an inverted index structure, where each term is mapped to a list of intervals, sorted on their $i\text{-}score$. Next, we discuss how we can use this index to evaluate multi-term queries.

Evaluation: First, we formally define the *Interval Evaluation Problem* as such:

Problem 5. Interval Evaluation Problem: *Given a query of terms $q = \{t_0, t_1, ...\}$, retrieve the k intervals with the highest values for $\sum_{t \in q} i\text{-}score(t, d)$.*

For the top-k evaluation phase, we introduce a modified version of the TA Algorithm, which we refer to as TA* (Algorithm (3)). TA* is similar to TA, differing only in the use of the random access probe. In the standard version, a random access probe looks for the candidate document in the various document lists and retrieves its $d\text{-}score$ (line 10 of Algorithm 2). In the case of intervals, this step is more complicated, since the candidate may overlap with multiple intervals in a list. Procedure (1) provides an implementation of the Random Access probe. Given an Interval I and a list of intervals L the probe returns **the set** of (sub)intervals of I that overlap with some interval in L. The procedure can be easily implemented with the use of interval-trees [24].

Procedure 1 RandomAccess(Interval I, Interval List L)

```
  Return a set of intervals $\mathcal{I}$, s.t.
  $\forall$ I' $\in$ L, where I' $\cap$ I $\neq \emptyset$,
  $\exists$I* $\in \mathcal{I}$, where I* = I' $\cap$ I  AND I*.score = I.score + I'.score
```

After the sets of overlapping (sub)intervals from each List have been retrieved, they are merged to produce the final set $\mathcal{F}$, consisting of segments included in bursty intervals for *all* the terms of the query (Line 11 of Algorithm 3). Figure (2.2) shows an example of the

Algorithm 3 TA* Algorithm

```
Input: query $q = \{t_0, t_1, ...\}$, int k
Output: Set of top-k Intervals
TopK $\leftarrow \emptyset$ // sorted, holds at most k distinct elements
Threshold T $\leftarrow$ 0
$\mathcal{L} \leftarrow \{L_0, L_1, ..\}$ // set of Doc Lists for each term in q
while (Not All lists in $\mathcal{L}$ Have Been Exhausted) do
    for (every List L $\in \mathcal{L}$) do
        cand $\leftarrow$ getNext(L)
        T $\leftarrow$ (T $-$ lastseen(L) $+$ cand.score)
        $\mathcal{X}$[i] $\leftarrow$ {cand}
        for (every List L' $\in \mathcal{L}$, L' $\neq$ L) do
            $\mathcal{X}$[j] $\leftarrow$ RandomAccess(cand, L')
        $\mathcal{F} \leftarrow$ merge($\mathcal{X}$[0], $\mathcal{X}$[1], ...)
        for every Interval I in $\mathcal{F}$ do
            TopK.insert(I, I.score)
        if ((TopK.size() == k) && (TopK.last() >= T))
            return TopK // Early Termination
return TopK
```

- getNext(L), lastSeen(L) and TopK.last() are as in the TA Algorithm.
- The $\mathcal{X}[\,]$ variables represent sets of intervals.
- The RandomAccess() function is described in Procedure 1.
- The use of the merge() function is shown in Figure (2.2).

merging process for a query $q = \{t_0, t_1, t_2, t_3\}$. The interval-set $\mathcal{X}[0]$ contains only one interval: the candidate, selected under sorted access from the bursty-interval list of term t_0. Also, $\mathcal{X}[1], \mathcal{X}[2]$ and $\mathcal{X}[3]$ contain intervals that overlap with the candidate, retrieved by applying the RandomAccess Procedure on the bursty-interval lists of terms t_1, t_2 and t_3, respectively. According to Definition 3, only the interval $I = [5:7]$ qualifies as **bursty with respect to q**. Following Eq. (2.14), the burstiness score of candidate I is equal to $\sum_{t \in q} B(t, super(\mathcal{I}_t, I)) = 6 + 4 + 5 + 4 = 19$.

The top-k set produced by TA* optimally solves the **Interval Evaluation Problem**. The reported intervals reveal bursty periods for any multi-term query. This allows us to not

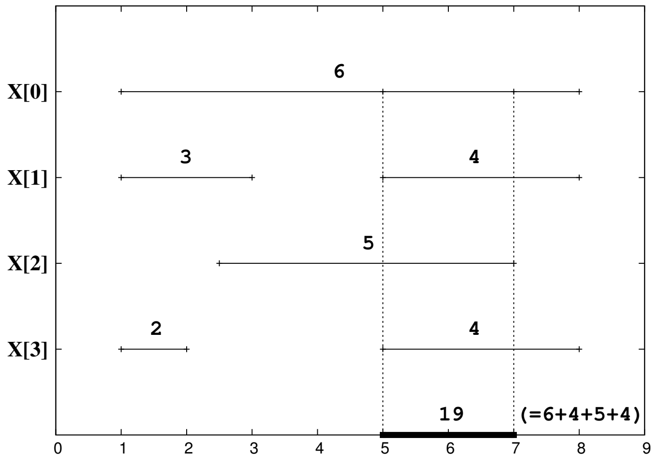


Figure 2.2: Interval Merging Process

only locate events correlated with particular terms, but also estimate their lifespan. Further, the framework can be easily extended to report the documents that appear within each interval, and also contain all the query terms. Thus, we can obtain ranked groups of documents, where each group is relevant to a specific bursty period. Clearly, this is more informative than a mechanism that simply reports k documents from completely arbitrary timestamps.

2.6 Experiments

In this section, we illustrate the efficacy of our search framework through a rigorous experimental evaluation. Section 2.6.1 describes the datasets we used. Section 6.2 discusses the different burst-detection methods used in our experiments. Finally, Sections 6.3-6.5 evaluate our search framework in different scenarios.

2.6.1 Datasets

Newspaper Datasets: We have conducted a series of experiments using real-world datasets from the Center for Bibliographical Studies and Research (CBSR) at the University of California, Riverside (UCR). CBSR has received two grants from the National Endowment for the Humanities to participate in the National Digital Newspaper Program (NDNP). The NDNP is a joint venture of the National Endowment for the Humanities and the Library of Congress to create a national digital newspaper resource, representing papers from all states, published between 1836-1922.

For the experimental evaluation we have gathered over 390,000 articles from the *San Francisco Call*, a daily newspaper with publication dates between 1900-1909. After the removal of stopwords, approximately 120,000 distinct terms were identified. We have several attributes for each article, including the title, the date of publication, and the raw (punctuation and capitalization included) content. Due to the age and size of the corpus, some issues were not located for digitization, leaving small gaps in the data set. To address this, we extracted 3 independent document sequences from the data, for which all the articles were available:

- `SF-Call-1`: A sequence of 122,114 articles spanning from Jan 01, 1900 to Dec 31, 1901.
- `SF-Call-2`: A sequence of 144,289 articles spanning from Jan 01, 1903 to Dec 31, 1904.
- `SF-Call-3`: A sequence of 153,412 articles spanning from Jan 01, 1908 to Dec 31, 1909.

These large sequences of chronologically ordered articles will serve as datasets for the experiments described in this section.

Major Events List: In order to perform a qualitative evaluation of our approaches, we manually composed a list of major events that took place at a time covered by one of the three **Newspaper Datasets**. The events were taken from Wikipedia (www.wikipedia.com), which maintains annual lists of major events. For every event, a query was composed, consisting of keywords chosen for their particular significance with respect to the event. Table (2.3) contains the list of events and their respective queries.

2.6.2 Burst Detection

Throughout the experiments section, we evaluate the performance of the proposed search frameworks, using the MAX-1 and MAX-2 algorithms, described in Section 2.4, to obtain the required bursty intervals. As an alternative, we try the popular burst-detection method proposed by Kleinberg in [12]. This algorithm is based on a Hidden Markov Model, with states that correspond to frequency levels for individual terms. State transitions (bursts) correspond to points in time, around which the frequency of a term changes significantly. Given the frequency sequence Y_t of a term t, dynamic programming is used to fit the most possible state sequence that is likely to have generated Y_t. The state assigned to each interval will serve as its burstiness score, which is required by our framework. For the rest of this paper, we refer to this algorithm as KLEIN.

The states reported by KLEIN form a hierarchical structure, with a long burst of low intensity including several bursts of higher intensity. Clearly, this violates our requirement for non-overlapping bursty intervals. To address this, we give priority to higher-state intervals, by assigning to every timestamp i the highest state observed over all the reported intervals that include i. To be fair, if the length of the highest-state interval is too small(¡3), we take the interval with the second-highest state. We believe this to be a reasonable and intuitive aggregation method.

Further, by assigning a high cost to state transitions, one can restrain the number of states in the hierarchy reported by KLEIN, thus eliminating short bursts and leading to longer intervals. Reasonably long intervals that reflect the true lifespan of an event are desirable, since they are likely to contain more relevant documents. On the other hand, the assignment of very high costs will limit the score-space to a small set of (low-intensity) states. Consider having to rank 10 documents based on their state, where each document has 1 of 2 distinct states; inevitably, multiple ties will lead to a meaningless ranking. For our experiments, KLEIN was tuned to find a balance between reasonably long intervals and an adequate number of distinct states. Note that our parameter-free algorithms resolve such issues by using the concept of the maximal segment to automatically extend a segment as long as it can benefit its score.

2.6.3 Document Ranking

The purpose of this experiment is to evaluate the Document Evaluation framework described in Section 2.5.1. The evaluation is done as follows: First, an inverted index is built on top of each on the three **Newspaper Datasets**, as described in Section 2.5.1. Then, the queries from the **Major Events List** are evaluated using the `TA` Algorithm. Queries mapped to events from 1900 and 1901 are evaluated using the index built on top of `SF-Call-1` and so forth. The entire process is repeated 3 times, each time using one of the three burst-detection algorithms (`MAX-1`, `MAX-2` and `KLEIN`) to build the search framework. We also compare against **Lucene** (lucene.apache.org), a popular text-search engine. Lucene uses frequency-based measures such as the frequency of the term within each document and the term's global frequency to rank documents in the context of a given query.

A human annotator studied each of the top-10 documents reported for each event, marking them as "relevant" or "non-relevant". This allows us to evaluate the achieved precision, defined as the ratio of the number of relevant documents over the total number of retrieved documents. The results are shown in Table (2.1). The table contains a separate column for the achieved recall in the top-5 documents, to provide more insight on the quality of the produced ranking. Our framework performs consistently well, clearly outperforming Lucene in almost every case. Regarding the different burst detection algorithms, `MAX-1` and `MAX-2` achieved near-perfect precision values for all submitted queries. `KLEIN`'s precision was just as good, although it failed to retrieve any documents for 5 of the 16 queries. This can be due to the fact that `KLEIN` did not identify any intervals as bursty for **all** the terms in the query. Alternatively, even if such a region was identified, it did not include any

Table 2.1: Achieved Precision on Major Events List

ID	**Lucene**		MAX-1		MAX-2		**KLEIN**	
1	1/5	2/10	5/5	10/10	5/5	9/10	5/5	8/10
2	3/5	5/10	5/5	7/10	5/5	8/10	5/5	8/10
3	3/5	6/10	5/5	10/10	5/5	10/10	-	-
4	3/5	7/10	5/5	10/10	5/5	10/10	5/5	10/10
5	1/5	2/10	3/5	6/10	4/5	8/10	-	-
6	4/5	6/10	5/5	9/10	5/5	9/10	5/5	10/10
7	3/5	6/10	5/5	10/10	5/5	10/10	4/5	9/10
8	3/5	4/10	5/5	10/10	5/5	10/10	4/5	9/10
9	5/5	9/10	5/5	10/10	5/5	10/10	5/5	9/10
10	0/5	1/10	5/5	10/10	5/5	10/10	5/5	10/10
11	5/5	10/10	5/5	10/10	5/5	10/10	5/5	10/10
12	4/5	8/10	5/5	10/10	5/5	10/10	-	-
13	3/5	6/10	5/5	10/10	5/5	10/10	5/5	8/10
14	4/5	7/10	5/5	8/10	5/5	7/10	-	-
15	3/5	5/10	4/5	8/10	5/5	9/10	5/5	7/10
16	2/5	2/10	5/5	9/10	5/5	9/10	-	-

documents containing all the query-terms. This could be addressed by separately tuning the parameters of the algorithm for each term. In practice, however, this is not desirable for obvious reasons.

2.6.4 Interval Ranking

The purpose of this experiment is to evaluate the Interval Evaluation framework described in Section 2.5.2. The experiment is similar to the one described in the previous Section. In this case, the index built on top of each of the **Newspaper Datasets** was the one described in 2.5.2, which considers term burstiness to index **intervals** rather than documents. Also, the TA* Algorithm was used to evaluate the queries from the **Major Events List**. For each event, we identify the interval among the reported 10 that is closest to the actual date of the event. We then report the start and end dates of that interval. The process is again repeated 3 times, each time using one of the three burst-detection algorithms. The results are shown in Table (2.4).

Both MAX-1 and MAX-2 produce reasonable intervals for the evaluated queries. As anticipated, MAX-2 gives tighter intervals, which commonly span a few days or weeks around the actual date of the event. The intervals produced by KLEIN are of similar or smaller length. Also, no bursty intervals were identified for queries 3, 5, 12, 14 and 16. As discussed in Section 2.6.2, even though the algorithm could be tuned to report larger segments, this would also reduce the number of states and thus have an adverse effect top-k evaluation. In general, KLEIN produced accurate results, indicating that our search framework is compatible with any efficient burst-detection method. Finally, it is also important to note that, for all three algorithms, the intervals closest to the actual event date were always ranked **first** in the top-10 list.

In order to illustrate the utility of the proposed Interval Evaluation Framework, we do an additional experiment: let $\mathcal{A}$ be the set of all articles within the interval reported by a query. Also, let $\mathcal{V}$ be the set of distinct terms appearing in the titles of the articles in $\mathcal{A}$. We then report the top-10 terms from $\mathcal{V}$, ranked in descending order on the number of titles they appeared in. For lack of space, we only report the results reported by MAX-2, since it produced the most reasonable intervals for all the queries in the **Major Events List**. The results, shown in Table (2.5), prove that the documents of a top-k interval can be used to identify terms that describe the underlying event. In the context of a search engine, these terms can compose an informative cloud that suggests insightful queries to the user.

2.6.5 Index Statistics

In this experiment, we show that, by focusing only on bursty intervals, we can greatly reduce the number of documents mapped to each term. This fact, combined with the high-quality results shown in the previous experiments, proves that our index structure is compact, while preserving all the useful information for each term.

First, we build the Document Evaluation framework, described in Section 2.5.1, for each of the three **Newspaper Datasets**. For each dataset, we compute the average number of documents mapped to a term. The process is repeated 3 times, once for each of the three burst-detection algorithms. We compare against Lucene, which essentially maps each term to all the documents that include it. A similar evaluation is done for the Interval Evaluation Framework, described in Section 2.5.2: for each term, we compute the percentage of the timeline (spanned by each collection) that is covered by bursty intervals. We then report the average over all terms. The results are shown in in Table (2). As can be seen from the Table, our framework achieves a significant reduction in the number of documents. As anticipated, MAX-2 and KLEIN result in higher reductions, since they generally produce smaller intervals. Further, only a small percentage (as low as 8%) of the timeline is covered by bursty intervals. Nonetheless, as illustrated by our previous experiments, these intervals provide all the information that our search framework needs to effectively evaluate queries.

Table 2.2: Statistics Table

	SF-Call-1	SF-Call-2	SF-Call-3
Avg. number of documents per term			
Lucene	124.45	119.3	112.4
MAX-1	85.6	83.5	74.9
MAX-2	73.5	75.2	63.2
KLEIN	72.35	74.9	72.7
Avg. covered timeline % per term			
MAX-1	0.27	0.24	0.27
MAX-2	0.09	0.08	0.08
KLEIN	0.1	0.12	0.14

2.7 Conclusion

In this work we explored how term burstiness can be used to enhance the search process for large document sequences. We provided a formal definition of burstiness and proposed efficient, parameter-free algorithms for the identification of bursty intervals for any given term. The main contribution of our work is an efficient search framework that considers term burstiness in the indexing and ranking process. We describe two alternative versions of our framework, and discuss how they can be useful to a user querying a document sequence. Finally, we thoroughly evaluated our approaches on a new dataset, in the context of different scenarios.

Table 2.3: Major Events List

ID	**Description**	**Date**	**Query**
1	Mormon Leader B. H. Roberts is refused a seat in the US Congress due to his polygamy.	Jan 17 1900	polygamy
2	The German passenger ship Saale, owned by the North German Lloyd, catches fire at the docks in Hoboken, killing 326 people.	Jun 30 1900	saale
3	King Umberto I of Italy is assassinated by Italian-born anarchist Gaetano Bresci.	Jul 29 1900	king assassination
4	A powerful hurricane hits Galveston, Texas killing about 8,000.	Sep 8 1900	texas disaster
5	Queen Victoria dies at the age of 81.	22 Jan 1901	victoria death
6	The Great Fire of 1901 begins in Jacksonville, FL .	May 3 1901	jacksonville
7	Serbian King Alexander Obrenovic and Queen Draga are assassinated.	Jun 11 1903	serbian kings
8	Pope Leo XIII dies. He is later succeeded by Pope Pius X.	July 20 1903	pope death
9	A fire at the Iroquois Theater in Chicago kills 600.	Dec 30 1903	theater disaster
10	The Great Baltimore Fire in Maryland destroys over 1,500 buildings in 30 hours.	Feb 7 1904	baltimore
11	Battle of Guru: British troops under Colonel Francis Younghusband battle with Tibetan Troops, marking the beginning of the British Expedition to Tibet	Mar 31 1904	guru
12	A fire aboard the steamboat General Slocum in New York City's East River kills 1,021.	Jun 15 1904	steamboat disaster
13	Eugen Schauman assassinates Nikolai Bobrikov, Governor-General of Finland	Jun 16 1904	finland governor
14	King Carlos I of Portugal and Prince Luiz are shot dead in Lisbon.	Feb 1 1908	carlos luiz
15	Louis Bleriot is the first man to fly across the English Channel in an aircraft.	Jul 25 1909	english channel
16	A 7.0 Richter scale earthquake destroys Messina, Sicily and rocks Calabria, killing over 75,000 people and living thousands homeless.	Dec 28 1909	italy homeless

Table 2.4: Predicted Intervals for Major Events

Event ID	**Actual Date**	MAX-1	MAX-2	**KLEIN**
1	Jan 17 1900	5 Jan - 3 Apr (1900)	5 Jan - 26 Jan (1900)	5 Jan - 23 Jan (1900)
2	June 30 1900	25 Jan - 12 Jul (1900)	1 Jul - 12 Jul (1900)	1 Jul - 12 Jul (1900)
3	Jul 29 1900	15 Jul - 19 Aug (1900)	30 Jul - 5 Aug (1900)	-
4	Sep 8 1900	3 Sep - 10 Mar (1900/01)	9 Sep - 6 Oct (1900)	10 Sep - 14 Sep (1900)
5	Jan 22 1901	5 Oct - 17 Mar (1900/01)	28 Dec - 8 Feb (1900/01)	-
6	May 3 1901	24 Apr - 29 Jul (1901)	27 Apr - 20 May (1901)	4 May - 23 May (1901)
7	Jun 11 1903	11 Jun - 25 Oct (1903)	12 Jun - 25 Jun (1903)	12 Jun - 19 Jun (1903)
8	July 20 1903	5 Jul - 4 Jan (1903/04)	7 Jul - 22 Jul (1903)	20 Jul - 22 Jul (1903)
9	Dec 30 1903	22 Dec - 20 Aug (1903/04)	31 Dec - 26 Jan (1903/04)	31 Dec - 17 Jan (1903/04)
10	February 7 1904	19 Jul - 20 Mar (1903/04)	5 Feb - 20 Feb (1904)	8 Feb - 20 Feb (1904)
11	Mar 31 1904	1 Apr - 6 Apr (1904)	3 Apr - 5 Apr (1904)	1 Apr - 6 Apr (1904)
12	Jun 15 1904	14 May - 4 Sep 1904 (1904)	16 Jun - 20 Jun (1904)	-
13	Jun 16 1904	20 Mar - 30 Oct (1904)	17 Jun - 31 Jul (1904)	20 Jun - 23 Jun (1904)
14	Feb 1 1908	2 Feb - 20 Feb (1908)	2 Feb - 11 Feb (1908)	-
15	Jul 25 1909	5 Mar - 10 Nov (1909)	19 Jun - 8 Aug (1909)	18 Jul - 27 Jul (1909)
16	Dec 28 1909	28 Nov - 28 Oct (1908/09)	26 Dec - 18 Jan (1908/09)	-

Table 2.5: Frequent keywords extracted from the top-k documents for each query

ID	Cloud
1	January state washington roberts present practice utah member house law
2	burn german york lloyd bodies fires recovered north river hoboken
3	humbert july state anarchist italiy unit rome bressi general police
4	city people galveston state sufferers received great money reported relief
5	state present great king queen people passed service palace city
6	city people fire state florida unite part sufferers generous report
7	belgrade queen peter officers alexander minister murder governor assassin palace
8	july leo rome cardinal holy pontiff church vatican present great
9	chicago fire place city building iroquois work time manager people
10	February state city general york fire company aid busy american
11	tibet british fight chinese mission general hostile colonel petersburg influence
12	bodies general york slocum fire boat hoboken police dead
13	general bobrikoff russia petersburg assassin government author people condition land
14	queen king crown assassination portugal prince oporto royal brother lisbon
15	flight july miles aviator attempt cross return bleriot condition machine
16	italian earthquake people city aid messina sufferers ruins relief stricken

Chapter 3

On the Spatiotemporal Burstiness of Terms

3.1 Introduction

The world wide web serves as a host to overwhelming volumes of documents, appearing in bulk online on a daily basis. Blogging and microblogging platforms (e.g. BlogSpot.com and Twitter.com), online magazines and newspapers (e.g. nytimes.com) and social networking platforms (e.g. Facebook.com) are examples of online venues where users flock to access such documents. In the context of such document streams, one of the most well-studied problem is the identification of bursts. Given a term t, a burst is generally identified when an unusually high frequency is observed for t in the posted documents. A significant amount of work has been devoted to identifying *temporal* bursts [25, 26]. A temporal burst is typically identified by: *a)* an interval on the timeline, indicating the specific timeframe during which the unusually high frequency was observed, and *b)* a score that indicates the burst's strength, i.e. the extent of the deviation from the term's usual frequency. The work on temporal burstiness assumes a single stream of documents. In the context of the web, however, documents are typically associated with a geostamp. In social networking plat-

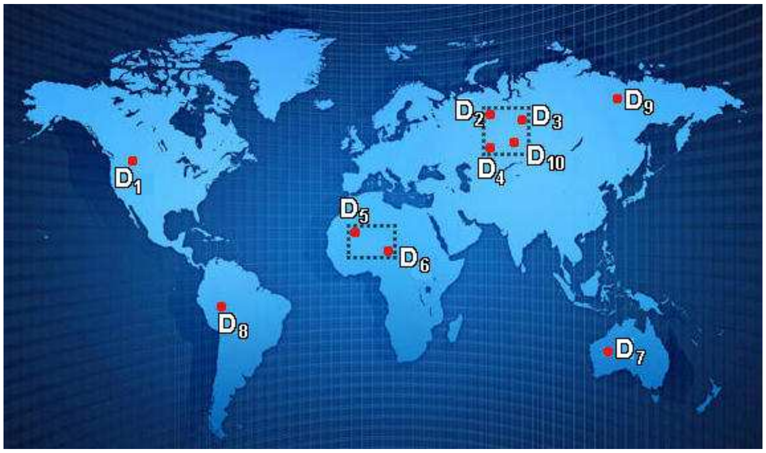


Figure 3.1: Spatiotemporal collection $\mathcal{D}$.

forms and blogging sites, registered users include their geographical location (i.e. place of recidence) as part of their online profile. Further, in news portals such as Topix.com articles are organized based on their place of origin. This setting motivates the study of burstiness in the spatial domain, by introducing multiple document streams from different locations. An example of this setting is shown in Figure 3.1, where the red dots on the map represent different document streams. In recent work, Mathioudakis et al.[27] have presented a framework for the identification of *spatial* bursts. In their work, the temporal interval of interest is given as part of the input (this is a limitation that we overcome in our work). Given such an interval I and a term t, the authors focus on identifying geographical regions where the observed frequency of t was unusually high, within the timeframe defined by I.

In this work, we present the first framework for simultaneously tracking the spatial and temporal burstiness of terms. In particular, given a set of document streams from different locations and a term t, we focus on two different types of *spatiotemporal burstiness patterns*:

- **Regional Patterns:** these patterns consider the geographical proximity among the document streams. They are defined as a combination of a temporal interval and a geographical region. A region can contain the geostamps (locations) of multiple document streams. Two such regions are marked in Figure 3.1. The first contains streams D_5 and D_6, while the second one streams D_2, D_3, D_4 and D_{10}. Conceptually, such a pattern encodes that *unusually high frequencies were observed for term t in geographical region R during a temporal interval I.*

- **Combinatorial Patterns:** these patterns ignore the geographical proximity among the streams. They are defined as combination of a temporal interval and a set of streams, where each stream originates from a different geographical location. Any arbirary subset of the streams marked in Figure 3.1 can be included in combinatorial pattern (e.g. $\{D_1, D_4, D_7\}$) Conceptually, such a pattern encodes that *unusually high frequencies were simulatenously observed for term t in all the streams in some set $\mathcal{C}$, during the same temporal interval I.* Note that $\mathcal{C}$ can contain streams from arbitrary locations.

In this work, we formalize both of these spatiotemporal patterns and present efficient algorithmic techniques for their identification.

Utilizing spatiotemporal burstiness: The second part of work focuses on the utilization of the mined spatiotemporal patterns. In previous work [25], we showed how temporal bursts can be used to identify documents on influential events. In this paper, we present the first search engine that considers the spatiotemporal burstiness of documents. Given a query of terms submitted by the user, our search engine retrievas relevant documents that discuss *events with a major spatiotemporal impact*, i.e. an impact that was reflected in multiple streams for an extended timeframe.

Not suprisingly, each of the two types of patterns described above leads to a different document-retrieval paradigm. While the first type leads to documents on events with a strong localized impact, the second type favors events with a more global effect. We demonstrate and discuss this further in our experiments.

3.1.1 Roadmap

The rest of this work is organized as follows: In Section 3.7 we review the related work; Section 3.2 provides background while in Sections 3.3 and 3.4 we describe the two alternative approaches for identifying spatiotemporal-burstiness patterns. The experimental evaluation appears in Section 3.6. We conclude in Section 3.8.

3.2 Preliminaries

Document Streams: We assume an underlying geographical map and a set of document streams $\mathcal{D} = \{D_1[\cdot], ..., D_n[\cdot]\}$. Here, $D_x[i]$ represents the set of documents reported from stream D_x at timestamp i. Each stream is associated with a fixed geographical location (geostamp). For the sake of simplicity, we assume a single streaming source per location (e.g. the aggregated content of all the available blogs or websites in a city). All streams span an ever-expanding timeline.

Granularity: Our approaches place no restrictions on the possible locations of the document streams. However, if the number of considered streams is overwhelming, it can potentially hurt the performance of the algorithms. This issue can emerge when millions of individual users (e.g. on Twitter) are considered as individual streams. Processing these users individually would be both costly and redundant. For most real-life applications, it is sufficient to consider a stream as an entire city or, at most, a specific neighborhood. Then, users can be easily grouped to form the corresponding aggreagate streams. Still, if one chooses an even finer granularity, it is preferable to define the problem in the context of the region of interest, and adopt it as the underlying map (instead of using the entire original map). An alternative way to group users is by using a grid to partition the underlying map. Each cell of the grid can then be considered as a different stream. **Spatiotemporal**

Patterns: We explore two different types of bursty spatiotemporal patterns: regional and combinatorial. Regional patterns are charaacterized by a region R of the geographical map, a timeframe I and a burstiness score. Note that R can contain multiple streams from $\mathcal{D}$. In the definition of combinatorial patterns, the region is replaced by an arbitrary subset of the streams in $\mathcal{D}$.

3.3 Combinatorial Patterns

In this section we introduce STComb, an approach for the identification of combinatioral spatiotemporal patterns. These patterns are defined as combination of a temporal interval and a set of streams, where each stream originates from a different location.

This approach builds upon our previous work [25], in which we showed how we can identify temporal bursts. Given a single stream of documents and a term t, we showed how we can extract, in linear time, the set of non-overlapping *bursty temporal intervals*.

Here, we extend this work in order to efficiently deal with *multiple streams* from different geographical locations. First, we use our previous method [25] to independently extract the sets of bursty temporal intervals for each stream. Note that, since the intervals reported for each stream are strictly non-overlapping [25], overlap can only exist between intervals from different streams. Each segment that exists in the ovelap of multiple intervals represents a spatiotemporal pattern, defined by the timeframe spanned by the segment and the set of locations where the overlapping intervals come from. Figure 3.2 shows examples of bursty temporal intervals for 4 document streams D_1, D_2, D_3 and D_4. For instance two intervals I_1 and I_2 have been identified for D_1, with their respective (temporal) burstiness scores being 0.8 and 0.5. Note that the temporal burstiness $\mathcal{B}_T(I)$ of an interval I is always a in $[0, 1]$.

Let $\mathcal{I}$ be the complete set of temporal intervals reported from all the document streams. Then, the problem of identifying spatiotemporal patterns is now translated into finding subsets of overlapping intervals. A subset $\mathcal{I}' \subseteq \mathcal{I}$ is eligible only if all the intervals it includes

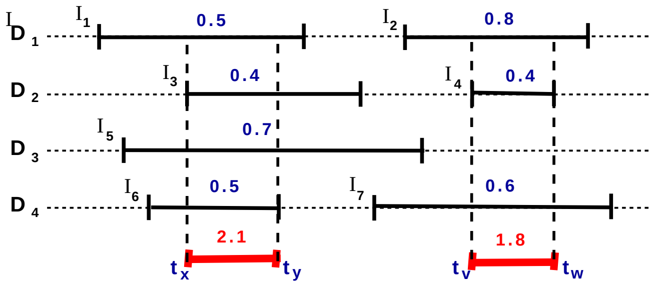


Figure 3.2: Examples of bursty temporal intervals for 4 document streams D_1, D_2, D_3 and D_4

.

share a common segment. In the example of Figure 3.2, we have $\mathcal{I} = \{I_1, I_2, I_3, I_4, I_5, I_6, I_7\}$. In this case, the subsets $\{I_1, I_3, I_5, I_6, \}$ and $\{I_2, I_4, I_7\}$. On the other hand, the subset $\{I_1, I_4, I_6\}$ is not eligible.

Formally, $\mathcal{I}'$ is eligible if:

$$\bigcap_{I \in \mathcal{I}'} I \neq \emptyset \tag{3.1}$$

To aid us in our analysis, we define $\mathcal{U}$ to be the universe of all eligible subsets of $\mathcal{I}$. First, we formally define the problem of finding the single highest-scoring subset of intervals:

Problem 6. Highest-Scoring Subset (HSS): *Let $\mathcal{U}$ be the set of eligible subsets, and let $\mathcal{B}_T(I)$ return the temporal burstiness score of a given interval I. Then, we want to find the subset $\mathcal{I}^* \in \mathcal{U}$ such that:*

$$\mathcal{I}^* = \underset{\mathcal{I}' \in \mathcal{U}}{\operatorname{argmax}} \sum_{I \in \mathcal{I}'} \mathcal{B}_T(I) \tag{3.2}$$

Solving the **HSS** problem gives us the highest scoring spatiotemporal pattern. Toward the end of this section we discuss how we can retrieve multiple high-scoring patterns. Note that any subset of intervals $\mathcal{I}' \in \mathcal{U}$ can be trivially converted into a (combinatorial) spatiotemporal pattern. A combinatorial pattern is defined by a set of streams, a timeframe and a burstiness score. Recall that, by the definition of $\mathcal{U}$, each interval in $\mathcal{I}'$ comes from a different stream. Therefore, all the streams that are represented (by a single interval) in $\mathcal{I}'$ compose the set of streams of the pattern. Further, the timeframe of the pattern is defined as the common segment of all the intervals in $\mathcal{I}'$. Finally, the burstiness score equal to $\sum_{I \in \mathcal{I}'} \mathcal{B}_T(I)$.

In the example of Figure 3.2, the highest scoring subset is $\{I_1, I_3, I_5, I_6\}$, which gives us the top spatiotemporal pattern. The set of streams included in the pattern is $D_1, D_2, D_3, D_4\}$. The burstiness of the pattern is 2.1, equal to the cumulative (temporal) burstiness of the included intervals. Finally, the timeframe of the pattern is defined by the common segment of the intervals, spanning from timestamp t_x to timestamp t_y in the figure.

Before we present our solution to the **HSS** problem, we state the following lemma, which will be useful in our further analysis:

Lemma 1. *Given a set $\mathcal{I} = \{I_1, ..., I_m\}$ of 1-D intervals on the real line, the following two statements are equivalent:*

$$\bigcap_{I \in \mathcal{I}} I \neq \emptyset \tag{3.3}$$

$$I_i \cap I_j \neq \emptyset,\ \forall (I_i, I_j) \in \mathcal{I} \tag{3.4}$$

Lemma 1 simply states that if m intervals have a non-empty intersection, then each pair of intervals must also have a non-empty intersection.

Given Lemma 1, we can now state the following Proposition:

Proposition 1. The **HSS** problem is equivalent to the *Maximum-Weight Clique Problem for Interval Graphs* (MWCI)

An instance of the Maximum-Weight Clique (MWC) problem consists of an undirected graph $G(V, E)$ and a vertex weight $w(v), \forall\, v \in V$. Given a constant K, the decision version of the MWC problem asks whether there exists a clique $V^* \subseteq V$, so that $\sum_{v \in V^*} w(v) \geq K$. Proposition 1 refers to a specialized formulation of this problem(MWCI), focusing exclusively on *Interval* Graphs. An interval graph is the intersection graph of a set of intervals on the real line. It has a vertex for each interval in the set, and an edge between every pair of vertices corresponding to two intersecting intervals. While MWC is known to be NP-Complete [28], MWCI is solvable in polynomial time [29].

Proposition 1 allows us to use any known algorithms for the MWCI problem to solve the **HSS** problem. In our experiments, we use the algorithm described in [29], which returns the single highest-scoring clique in $O(n \log n)$ time. We refer to this algorithm as `maxClique`.

Getting Multiple Patterns: In order to obtain multiple non-overlapping patterns we can iteratively apply `maxClique` and then remove the intervals included in the maximum clique. Allowing overlap would inevitably lead to uninformative results, obtained by trivially modifying other high-scoring cliques. Nonetheless, one can alternatively use any of the available algorithms for the enumeration of overlapping maximal cliques for interval graphs [30].

3.4 Regional Patterns

In the previous section we considered the combinatorial problem of finding sets of streams from different locations that exhibit bursty behavior on the same term for extended timeframes. While this approach produces great results, it is not appropriate for streaming data, since it needs to recompute the set of cliques every time new information arrives. In addition, `STComb` disregards the spatial proximity of the streams. Next, we describe an online approach, called `STLocal`, that addressed these issues. By considering the geographical proximity of the streams, we can evaluate the spatial extent of a term's burstiness pattern. Conceptually, we are looking for *bursty regions of the map*, instead of *arbitrary sets of bursty streams*.

First, we examine the case where we are given $D_x[i]$: the set of documents received from a single data stream $D_x \in \mathcal{D}$ at timestamp i. We then extend our approach to deal with a snapshot *of the entire collection,* taken at some fixed point in time. Finally, we address the streaming scenario, where a new snapshot is added at every new timestamp.

Single Data Stream: We model spatiotemporal term burstiness using the formal concept of *Discrepancy*. Discrepancy Theory has different formalizations and applications in several fields [31] and is generally used to describe the deviation of an *observed* situation from the *expected* baseline. Next, we use this paradigm to model the burstiness of a given term t: let $D_x[i]$ represent the set of documents that arrived from a stream $D_x \in \mathcal{D}$ at timestamp i. Then, given a term t, let $D_x[i][t]$ return the total frequency of t in the documents included in $D_x[i]$. Formally:

$$D_x[i][t] = \sum_{d \in D_x[i]} \mathit{freq}(t, d) \tag{3.5}$$

$D_x[\cdot][\cdot]$ can be visualized as a 2-D matrix, where rows correspond to timestamps and columns to terms. Then, $D_x[i][t]$ represents the frequency that was *observed* for t on timestamp i.

Following the typical Discrepancy paradigm, we now define $E_x[i][t]$ to be the *expected* frequency of t with respect to stream D_x at timestamp i. This allows us to identify and evaluate frequency bursts by measuring the extent to which the *observed* frequency surpasses the *expected* baseline. The nature of an appropriate baseline depends on the domain of the application and the specifics of the data: $E_x[i][t]$ can be taken to be equal to the average observed frequency of t in D_x, taken over all the snapshots collected before timestamp i. Alternatively, one can focus only on the most recent measurements. Finally, data from previous timeframes can also serve as a baseline, if available. For example, the expected frequency of a given term t in the news from *San Francisco* on Dec-25-09 can be computed as the average daily frequency of the term, as computed over the measurements taken during the Dec. of previous years. We define the burstiness of a given term t with respect to a data stream $D_x \in \mathcal{D}$ at timestamp i as follows:

$$\mathcal{B}(t, D_x[i]) = D_x[i][t] - E_x[i][t] \tag{3.6}$$

Algorithm 4 R-Bursty

```
**Input:** term $t$, snapshot $\mathcal{D}_i$ of a spatiotemporal collection $\mathcal{D}$
**Output:** All **non-overlapping** rectangles in $\mathcal{D}_i$ that have $r\text{-}score(\cdot, i, t) > 0$.
1: Run the algorithm in [31] to retrieve $R_{max}$, the rectangle in $\mathcal{D}_i$ with the highest *r-score*.
2: Report $R_{max}$ and set $\mathcal{B}(t, \mathcal{D}_x) = -\infty, \forall D_x \in R_{max}$
   ( We set the scores of the streams within $R_{max}$ to $-\infty$ to eliminate overlap among the reported rectangles).
3: Repeat the process from the first step, until the *r-score* of the retrieved rectangle is less or equal to zero.
```

Snapshot of the Entire Collection: A snapshot $\mathcal{D}[i] = \{D_1[i], D_2[i], ..., D_n[i]\}$ of a spatiotemporal collection $\mathcal{D}$ consists of the document-sets reported by *all* the streams at a single timestamp i. `STLocal` considers the spatial locality various streams in the 2-D space: we want to find *regions* that are bursty with respect to a given term t. The burstiness of a region is based on the stremas that originate from within its area. Ideally, we could afford the flexibility of looking for regions of arbitrary shapes. However, this would dramatically increase the computational cost. Therefore, we focus on regions that can be represented by axis-oriented rectangles, allowing, as we show later, for a polynomial-time solution of the problem. By allowing rectangles of arbitrary size, we can capture interesting patterns on the 2-D map, while achieving an acceptable computational cost. A rectangle may contain multiple streams, depending on its size and location on the map. In the example of Figure 3.1, the rectangular area in north Africa includes streams D_5 and D_6.

We define the rectangle score ($r\text{-}score$) of a rectangle R with respect to a term t at a given timestamp i as the sum of the respective burstiness values of the streams that fall within R. Formally:

$$r\text{-}score(R, i, t) = \sum_{D_x \in R} \mathcal{B}(t, D_x[i]) \tag{3.7}$$

where $\mathcal{B}(t, D_x[i])$ is as defined in Eq. 3.6. We can now formalize the notion of *Bursty Rectangles* as follows:

Definition 4. [*Bursty Rectangles*]: Given a term t and a snapshot $\mathcal{D}[i]$ of a spatiotemporal collection $\mathcal{D}$, we define as *Bursty Rectangles* the complete set of **non-overlapping**

rectangles, for which $r\text{-}score(\cdot, i, t) > 0$.

Positive-scoring rectangles represent regions where the overall observed frequency was higher than the expected one. The no-overlap constraint bounds the number of rectangles to at most $n = |\mathcal{D}|$. It also eliminates trivial results, produced by slightly modifying other high-scoring rectangles. In some cases a higher *r-score* can be achieved by expanding the rectangle to include more streams, even if it means also including some non-bursty streams. Our approach automatically determines whether a set of streams should be included in a single rectangle, or if reporting a set of (two or more) smaller rectangles would benefit the *r-score*. In Algorithm 4, we introduce the pseudo code of R-Bursty, an optimal algorithm to find *Bursty Rectangles*, that returns *all* non-overlapping rectangles that have a positive *r-score*. The R-Bursty algorithm uses the polynomial algorithm proposed in [31] to find the single axis-oriented rectangle with the maximum bichromatic discrepancy in a 2-D setup.

Complexity of R-Bursty: The complexity of the first step is $O(n^2 \log n)$ [31]. Since the number of non-overlapping rectangles is bounded by $n = |\mathcal{D}|$, the complexity of R-Bursty is $O(n^3 \log n)$. This polynomial cost becomes even more satisfactory if one considers that the number of streams n is typically limited (i.e. in the tenths or hundreds).

Streaming Data: The R-Bursty algorithm provides us with the set of bursty rectangles for a single snapshot of the collection. As new snapshots arrive in a streaming fashion, we want to aggregate the consecutive rectangle-sets, in order to identify extended periods of time when particular regions of the map displayed bursty behavior. To assist us with the analysis, we define the concept of the *spatiotemporal window* $w = (R, [a : b])$, consisting of an axis-oriented rectangle R in the timeframe $[a : b]$. Geometrically, a spatiotemporal window w can be represented as a hyper-rectangle in 3-D space. Figure 3.3 shows 3 different examples of spatiotemporal windows, w_1, w_2 and w_3, on a 60×40 map. Window w_1 corresponds to the rectangle R on the map, and spans the timeframe between 3 and 8. Also, observe that w_2 and w_3 correspond to the same rectangle, even though they span different

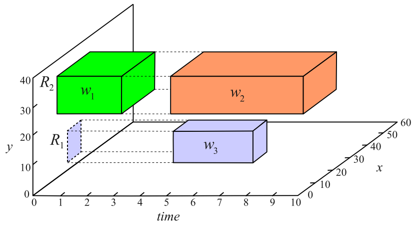


Figure 3.3: Example of Spatiotemporal Windows.

timeframes. Given a term t, we define the windows score (*w-score*) of spatiotemporal window $w = (R, [a : b])$ with respect to a term t as follows:

$$w\text{-}score(w,t) = \sum_{i=a}^{b} r\text{-}score(R,i,t) = \sum_{i=a}^{b} \sum_{D_x \in R} \mathcal{B}(t, D_x[i]) \tag{3.8}$$

Next, we show how we can use Eq. 3.8 to identify meaningful high-scoring spatiotemporal windows. First, let us formalize the concept of a *maximal* spatiotemporal window:

Definition 5. [*Maximal Spatiotemporal Window*]:

Given two windows $w = (R, [a : b])$ and $w' = (R', [a' : b'])$, we say that w' is a *sub-window* of w if w' is completely contained in w (in terms of both space and time, i.e., $R' \subseteq R$, $b' \leq b$ and $a' \geq a$). Thus, w is then considered a *super-window* of w'. Then, a window w is considered *maximal* if and only if there exist no super-windows of w that have a higher *w-score* than it does.

A maximal window represents a meaningful and informative spatiotemporal pattern. Given this concept, we formalize the Bursty Source Patterns problem by mapping it to the problem of finding the set of *Maximal Windows*. Given a spatiotemporal collection $\mathcal{D}$ and a term t, we want to find the set of positive-scoring, maximal spatiotemporal windows $\mathcal{W}_t$. A positive score means that, within the region covered by the window, the observed frequency of the term was higher than the expected one. In the context of streaming data from multiple streams, computing and maintaining W_t is a non-trivial task. In Algorithm 5, we show STLocal, an efficient algorithm that finds the *Maximal Windows*. The algorithm is polynomial in the number of document streams.

In certain cases, a window w may eventually lose its maximality due to another, higher scoring window from a different region that either contains or is contained in the region corresponding to w. Since data arrives in a streaming fashion, there is no way to predict such cases. However, basic bookkeeping can be employed to deal with such cases as they occur, without affecting computational complexity.

Given such a sequence of real values, we need an online process able to maintain $\mathcal{W}_t$. For this, we employ the algorithm presented in [23], which we refer to as GetMax. Given a sequence of real values, GetMax identifies all the maximal segments (i.e. contiguous subsequences) in linear time. Each maximal segment corresponds to a maximal window. In Line 10, GetMax is used to update the set of maximal windows $\mathcal{W}_t$ for the term.

Algorithm 5 STLocal

```
    Input: Spatiotemporal collection $\mathcal{D}$
    Output: Set of Maximal Windows $\mathcal{W}_t$ for every term $t$
1:  $i \leftarrow 0$ // Timestamp Counter
2:  Initialize $\mathcal{S}_t \leftarrow \emptyset$, $\mathcal{W}_t \leftarrow \emptyset$ for every term $t$
    ($\mathcal{S}_t$ contains a sequence of snapshots for every rectangle)
3:  while Stream is open do
4:      $i \leftarrow i + 1$
5:      for each term $t$ do
6:          $\mathcal{R} \leftarrow$R-Bursty($\mathcal{D}_i, t$)
7:          $\mathcal{S}_t \leftarrow \mathcal{S}_t \cup \{$new sequence $S : \forall R \in \mathcal{R}\}$
8:          for ( each sequence $S \in \mathcal{S}_t$) do
9:              $S$.add(r-score($R_S, i, t$))
10:             $\mathcal{W}_t \leftarrow \mathcal{W}_t \cup$ GetMax($S$)
11:             if ($S.total < 0$) then
12:                 Remove $S$ from $\mathcal{S}_t$
```

Complexity of STLocal: Since each term is processed independently, the process can be easily parallelized. The complexity is then as follows: let $|L|$ be the length of the timeline spanned by our collection. STLocal applies the R-Bursty algorithm $|L|$ times, thus requiring $O(|L|n^3 \log n)$, where $n = |\mathcal{D}|$ is the number of streams. Further, the maximum number of sequences (i.e. bursty regions) that need to be maintained is $O(n|L|)$. As we show in the experiments, the actual number is a lot smaller, since bursty artifacts are, by definition, rare. By using GetMax, we can maintain each window in $O(|L|)$ time, for a total of $O(n|L|^2)$. Therefore, the overall complexity is $O(|L|n^3 \log n + n|L|^2) = O(|L|n^3 \log n)$.

3.5 Searching for Bursty Documents

In the previous two sections we presented two alternative approaches for the extraction of bursty spatiotemporal patterns. Next, we show how we can use these patterns to retrieve documents that are relevant to a user's query and also discuss events with a high spatiotemporal impact. We refer to these documents as *bursty documents*. Even though our search engine is compatible with both regional and combinatorial patterns, it only handles one type at a time (i.e. a separate instance of the framework is required for each type).

On a high-level, our search engine considers two factors in the evaluation of a given document: 1) the relevance of the document to the user's query, and 2) the document's burstiness, as captured in its overlap with the reported spatiotemporal burstiness patterns. Formally given a query of terms q, the score of a document d is computed as follows:

$$\textit{score}(q,d) = \sum_{t \in q} \textit{relevance(d,t)} \times \textit{burstiness}(d,t) \tag{3.9}$$

Here, *relevance(d,t)* is the relevance of document d with respect to term t. This can be implemented as any normalized version of *freq(t,d)*, i.e. the number of occurrences of t in d. The best choice depends on the particular nature of the considered documents. In our own experiments, we found that using $\log($*freq(t,d* $+ 1))$ yielded the best results.

Further, *burstiness(d,t)* is the burstiness of document d with respect to term t. This depends on the overlap of the document with the spatiotemporal patterns that have been extracted for t. Let $\mathcal{P}_t$ be the set of patterns extracted for a given term t. Recall that both types of spatiotemporal patterns discussed in this work (combinatioral and regional) include a timeframe and a set of streams. In addition, each document d arrives from a single stream at a specific point in time. We say that d *overlaps* with a pattern P if both its stream of origin and its timestamp are included in P. Both of the approaches we discussed in the previous sections allow for overlap among the reported patterns. In that case, it is possible for a document to overlap with multiple patterns. Formally, let $\mathcal{P}_{t,d} \subseteq \mathcal{P}_t$ be the subset of the patterns reported for term t that overlap with a given document d. Then, we define the burstiness of d with respect to t as follows:

$$\textit{burstiness(d,t)} = \begin{cases} f(\mathcal{P}_{t,d}) & \text{if } \mathcal{P}_{t,d} \neq \emptyset \\ -\infty & \text{otherwise} \end{cases} \tag{3.10}$$

where $f(\mathcal{P}_{t,d})$ can be any function of the scores of the patterns $\mathcal{P}_{t,d}$. For example, $f(\cdot)$ can return the maximum, minimum or median such score. An aggregate function that considers all the scores, such as the average, can also be applied. In our own experiments, we found that using maximum score over all the patterns included in $\mathcal{P}_{t,d}$ yielded the best results.

Given Eq. 3.9, we can now formulate the *Bursty Documents* problem:

Problem 7. [*Bursty Documents*]: Given a set of streams $\mathcal{D}$ and a query of terms $q = \{t_0, t_1, ...\}$, we want to find the *k* documents from $\mathcal{D}$ with the highest burstiness, i.e. those the k documents that maximize Eq. 3.9

The problem can now be addressed via standard information-retrieval techqnques. An inverted index is first built, mapping each term to the documents that include it, ranked by their repspetived scores. The popular Threshold Algorithm (TA) [32] for top-k evaluation can then be applied to retrieve the top documents for any given mult-term query.

3.6 Experimental Evaluation

We proceed with an evaluation of our two proposed frameworks using two datasets:

Topix Dataset: For lack of an openly available dataset of proper sequences (i.e. with consecutive timestamps) of documents from different geographic locations, we composed a corpus of web-articles from [33], which hosts news-stories from different countries around the world. This dataset contains 305,641 articles, where the vast majority of them come from local news sources from 181 different countries, posted between Sep-08 and Jul-09. To project the sources' locations on the 2D plane, we use Multidimensional Scaling given the pair-wise geographical distances of sources using [34].

Major Events List: We composed a list of influential real-life events that took place during the timeframe spanned by the dataset. The events were taken from [35], which maintains a list of major events for every calendar year. We identify three loosely-defined categories of events in the list: events with a significant global impact (events 1–6), major events that were reported in a large number of countries (7–12) and events with a more localized impact (13–18). A short description of the selected events is given in Table 3.1. Each event was shown to a human annotator, who was instructed to provide the query that would be submitted to a search engine, if looking for information on that event.

3.6.1 Bursty Source Patterns Evaluation

In this experiment we evaluate the two proposed approaches in the context of the Bursty Source Patterns problem. We use our two approaches to retrieve the top-scoring bursty source pattern, given each of the queries from the Major Events List. Table 3.2 shows the number of countries included in the top pattern by `STComb` and `STLocal`. For `STComb`, we also report the number of countries included in the Minimum Bounding Rectangle (MBR) of the set of countries included in the top clique. This illustrates the different ways in which the two algorithms consider the spatial information of the data.

Table 3.2 provides valuable insight on the behavior of the two algorithms. For events with a global impact (e.g. the death of singer Michael Jackson or the global financial crisis), both `STLocal` and `STComb` report large spatiotemporal patterns, covering the majority of the available data sources. For the events of the middle tier (e.g. the acts of piracy in Somalia), the results of the two approaches begin to differ, with `STComb` generally

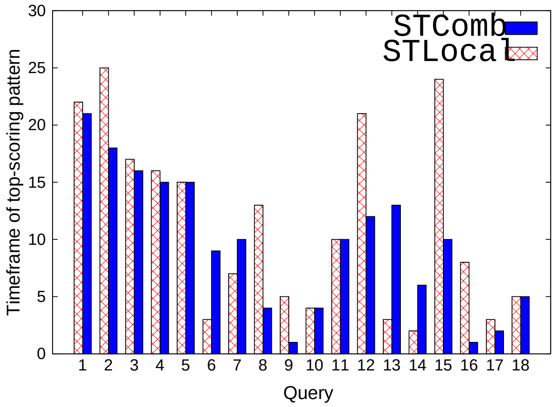


Figure 3.4: Timeframe length of the top pattern for queries in the Major Events List.

including more countries in the top pattern. This difference becomes even more apparent for events with a more localized impact (e.g. the inauguration of M. Tsvangirai as the new Prime Minister of Zimbabwe). For such events, `STLocal` reports small patterns that focus on the area around the event's source. On the other hand, `STComb` reports larger patterns with countries from around the globe. These patterns were often many times bigger than the respective ones given by `STLocal`.

This behavior was anticipated since `STLocal` is bounded by the geographical proximity of the various data sources, thus grouping together countries that are both bursty and close to each other. On the other hand, `STComb` focuses exclusively on the maximization of burstiness, resulting in larger patterns with numerous sources from arbitrary locations on the map. This is also demonstrated by the sets of countries included in the MBR of the various patterns. These were very large sets that consistently included the vast majority set of the available sources.

We complete our analysis with the timeframes of the reported patterns, plotted in Figure 3.4: each pair of bars corresponds to a query, following the same order as in Table 3.2. The left bar of the par represents the timeframe spanned by the pattern given by `STLocal` and the right one the respective timeframe given by `STComb`. The y-axis represents the length of a timeframe in weeks. For most queries, the two approaches report timeframes of a similar length. There are cases, however, when `STLocal` reports longer timeframes.

This happens for events that stay in the spotlight in the area around their origin, even after the event dies out in locations further from the source.

In conclusion, the two algorithms fulfill the purposes for which they were designed: STLocal can track the spatiotemporal impact of events, which is especially meaningful for events that affect specific regions. On the other hand, STComb can be used to identify all the affected locations, regardless of their geographical coordinates.

3.6.2 Bursty Documents Evaluation

In this experiment we evaluate the two proposed approaches in the context of the Bursty Documents problem. Given the set of events from the Major Events List and their respective queries, we use the STLocal and STComb to retrieve the top-10 documents for each event. The retrieved documents are then given to a human annotator, who marks each of them as "relevant" or "not relevant" to the event. This allows us to evaluate the *precision* of the two approaches.

We compare the results with the search engine we described in [25], which focuses exclusively on the *temporal* burstiness of terms. We refer to this approach as TB. For TB all the documents from the various countries where merged to a single set, since this approach disregard the origin of each document.

The three approaches consistently reported high precision, as shown in Table 3.3. STLocal was perfect for all queries and STComb for all except one (Q_{13}, with 80% precision). TB had a few false positives for the events in the 3rd category (i.e. the ones with a more localized impact), with an average of 80% precision. This can be explained by the fact that TB focuses only on the global maximization of the temporal burstiness, assuming a single source. Therefore, TB can be less sensitive to events with a more limited, localized impact.

To perform a more thorough analysis of the results, we explore the similarity between the top-k sets reported by the approaches. A characteristic example is the query "earthquake": all 10 documents returned by STLocal discussed the 2009 Costa Rica *Cinchona*

Earthquake. This was anticipated, since the algorithm considers the geographical locations and proximity of the sources on the map. Among the documents given by `STComb`, 3 were on the Sichuan earthquake in China, 3 were on an earthquake in Guerrero, Mexico and all the others discussed earthquakes from different countries across the world. Finally, for `TB`, 3 articles were on the same earthquake from Bulgaria, while all others discussed different locations. To quantify the difference between the algorithms, we calculate the similarity (defined as the size of the overlap divided by 10) between their top-k sets. The similarity values where 0.61 for `STComb`-`TB`, 0.58 for `STComb`-`STLocal` and 0.67 for `TB`-`STLocal`. This raises an interesting point: even though all 3 algorithms have an extremely high precision, their top-k sets can differ significantly. By optimizing different facets of burstiness, the 3 approaches report diverse results and complement each other.

3.6.3 Performance Evaluation

The complexity of the `STLocal` algorithm is $O(|L|n^3 \log n)$, where n is the number of data sources and $|L|$ the length of the stream (i.e. number of timestamps). This worst-case complexity assumes that, for a given term t, there exist $O(n)$ bursty rectangles in every 2-D snapshot taken at a single timestamp. However, in practice, the number is a lot smaller than n. We evaluate this on the Topix dataset, for which n (number of countries) is equal to 181. First, we compute the average number of bursty rectangles reported for each term per timestamp. We then build a histogram of the computed population. The results show that, for the vast majority of terms (92%), the average number of rectangles per timestamp was between 0 and 1, far smaller than the 181 assumed by the worst-case scenario. The number of rectangles is between (1-3] for 4% of terms, (2-3] for 3% of terms, and $\geq$ 3 for only 1% of terms.

Another factor that affects the complexity of `STLocal` is the number of spatiotemporal windows that need to be maintained. The worst-case analysis assumes that, for a timeline of length L, this number is $O(n|L|)$ (i.e. n new windows per timestamp). As we show

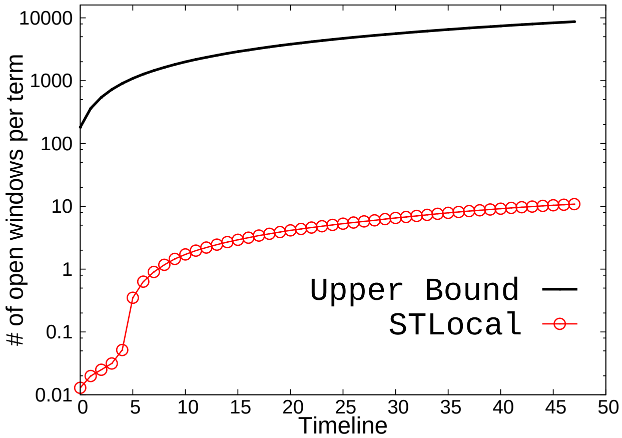


Figure 3.5: Number of open spatiotemporal windows.

using the Topix dataset, the number in practice is considerably smaller. For this dataset, $n|L|$ translates to a total of 181×48=8,688 distinct windows. The total number of open windows per time instance, as reported by `STLocal`, is shown in Figure 3.5. The number shown is the average taken over all the terms in the collection. We also plot the worst-case number for each timestamp (181 for timestamp i=1, 362 for i=2, etc.). The number assumed by the worst-case scenario is several orders of magnitude larger than the one observed for real data, reaching a maximum at around 10 open windows per term.

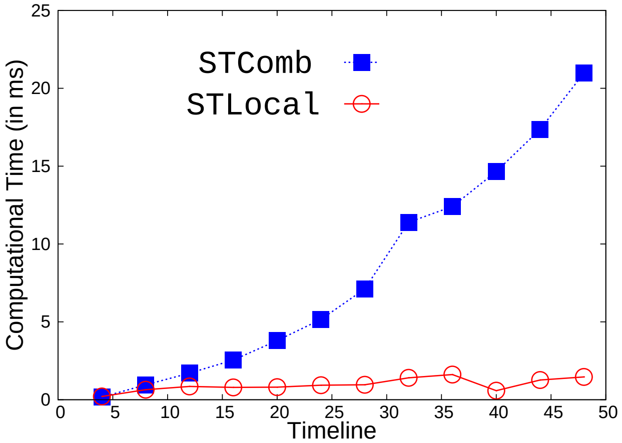


Figure 3.6: Running time (ms) per timestamp.

We conclude our study with a comparison of the computational time required by our proposed algorithms to process the Topix dataset. Our experiment emulates the streaming scenario, i.e., we process the collection one timestamp at a time, in sorted order by timestamp. Since the processing of each term is independent for both algorithms, we report the average time required to process a single term in each timestamp. Figure 3.6 shows that `STLocal` clearly outperforms `STComb`. This was anticipated, since `STLocal` is an online algorithm, with the ability to update the information for each term, every time new data arrives. On the other hand, `STComb` needs to be re-applied to the entire updated dataset. `STLocal` consistently required times around 1ms, exhibiting great performance and scalability. That being said, it is important to note that the results for the `STComb` are encouraging: even when asked to process the entire stream, the algorithm required as little as 20ms per term. This illustrates the potential of the `STComb` and motivates us to work on an online version of the algorithm.

3.7 Related Work

A number of works explore the spatiotemporal aspects of textual collections, albeit in a different context. For instance, [36] gives an overview of a pixel-based approach for the visualization of spatiotemporal events discussed in microblogging sites. [37] describes a system for large-scale analysis of blogs and online news. In [38], the spatiotemporal dimension of the data is explored to identify clusters representing emerging trends. In [39], a clustering technique that uses the users' locations and the content of the user's comments (tweets) on Twitter is proposed to identify locations of topics. Our spatiotemporal paradigm differs significantly from all these proposals, as we mine spatiotemporal burstiness of terms from streams originating in different locations.

Related to our problem formulation is the work in [40], that addresses spatiotemporal *theme mining* on blogs. However, our setup is different in many ways. First, [40] focus primarily on pattern mining, while we focus on search. [40] considers the spatiotemporal aspects of a given set of *themes* (topics), while we are interested in the spatiotemporal burstiness of terms to build a search engine for finding documents on influential events (or topics) that are relevant to a textual query. Second, their approach, unlike ours, does not account for streaming data. Our problem formulation is dynamic in the way it considers both spatial and temporal information: given a set of terms, it asks for sets of bursty locations, or regions of the map, that were bursty for extended timeframes. In other words, both the locations and the timeframe are identified automatically. In contrast, [40] finds the *life cycle* (timeframe) of a given theme, without reporting sets of bursty locations or map regions. In addition, for a given theme, the spatial dimension is only considered for a single fixed timestamp, for which the map of distributions over all locations is returned.

[41] presents a spatiotemporal analysis of relevant feeds by using a binary SVM to classify Twitter feeds as relevant or non-relevant to a given event. This introduces the need for training data, contrary to our completely unsupervised approach. [42] describes BlogScope, a search engine for blogs. While temporal information is considered, the spatial dimension is adopted in a very basic manner, allowing the user to select a specific region of the map to view data and analytics. Instead, we want to simultaneously maximize the spatiotemporal burstiness of terms and automatically identify regions of the map that are bursty with respect to a term for extended timeframes. [27] is an extension of [42], where the goal is to find spatial bursts in *fixed* temporal interval in a grid-based spatial layout. Our approach is more general in the sense that it can simultaneously track spatial and temporal burstiness, without the drawback of being tied to a grid structure with fixed-cell size.

We note that our context differs in numerous ways from previous work on querying, indexing and mining spatiotemporal data [43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]. In addition to our modeling of term burstiness instead of moving objects, our queries are purely textual and make no use of spatial or temporal predicates.

3.8 Conclusion

In this work we formalized the spatiotemporal burstiness of terms and showed how it can be measured and utilized toward an efficient search engine. Our engine returns documents on influential events with a major spatiotemporal impact. We proposed two alternative approaches, `STComb` and `STLocal`. The two approaches are complementary, providing valuable insight on spatiotemporal burstiness from different perspectives. Finally, we demonstrated the efficacy and efficiency of our methods through a rigorous experimental evaluation on real data.

Table 3.1: List of Major Events between September of 2008 and July of 2009, from `www.wikipedia.com`.

#	Query	Event Description
1	Obama	Events regarding the actions of B. Obama, the new President of the USA since January of 2009
2	financial crisis	Events regarding the global financial crisis
3	terrorists	Events regarding terrorism
4	Jackson	American entertainer Michael Jackson passes away.
5	swine	Events regarding the 2009 swine flue pandemic
6	earthquake	Events regarding earthquakes
7	gaza	Events regarding the Israeli Palestinian conflict in the Gaza Strip
8	ceasefire	Israel announces a unilateral ceasefire in the Gaza War
9	yemenia	Yemenia Flight 626 crashes off the coast of Moroni, Comoros, killing all but one of the 153 passengers and crew
10	Air France	Air France Flight 447, en route from Rio de Janeiro to Paris, crashes into the Atlantic Ocean, killing all 228 on board
11	piracy	Events regarding incidents of Piracy off the Somali coast
12	bush fires	Deadly bush fires in Australia kill 173, injure 500 more, and leave 7,500 homeless.
13	Nkunda	Congolese rebel leader L. Nkunda is captured by Rwandan forces
14	Vieira	The President of Guinea-Bissau, J. B. Vieira, is assassinated
15	Tsvangirai	M. Tsvangirai is sworn in as the new Prime Minister of Zimbabwe
16	Rajoelina	Andry Rajoelina becomes the new President of Madagascar after a military coup d'etat
17	Fujimori	Former Peruvian Pres. Fujimori is sentenced to 25 years in prison for ordering killings and kidnappings by security forces
18	Zelaya	The Supreme Court of Honduras orders the arrest and exile of President M. Zelaya

Table 3.2: Top-Scoring Bursty Source Patterns.

#	Query	# countries	# countries	# countries
		in STComb	in STLocal	in MBR
1	Obama	136	176	181
2	financial crisis	113	159	181
3	Jackson	151	132	181
4	terrorists	98	126	167
5	swine	157	174	181
6	earthquake	81	17	171
7	gaza	116	174	179
8	ceasefire	52	36	156
9	Yemenia	21	19	125
10	Air France	67	50	179
11	piracy	39	24	174
12	bush fires	30	3	168
13	Nkunda	22	30	118
14	Vieira	22	15	114
15	Tsvangirai	24	4	123
16	Rajoelina	30	4	154
17	Fujimori	19	5	158
18	Zelaya	55	26	171

Table 3.3: Precision in top-10 documents.

#	Query	TB	STComb	STLocal
1	Obama	1	1	1
2	financial crisis	1	1	1
3	terrorist	1	1	1
4	Jackson	0.9	1	1
5	swine	1	1	1
6	earthquake	1	1	1
7	gaza	1	1	1
8	ceasefire	1	1	1
9	Yemenia	1	1	1
10	Air France	1	1	1
11	piracy	1	1	1
12	bush fires	1	1	1
13	Nkunda	0.7	0.8	1
14	Vieira	0.8	1	1
15	Tsvangirai	0.9	1	1
16	Rajoelina	0.7	1	1
17	Fujimori	0.8	1	1
18	Zelaya	1	1	1

Chapter 4

Efficient Confident Search in Large Review Corpora

4.1 Introduction

Item reviews are a vital part of the modern e-commerce model, due to their large impact on the opinions and, ultimately, the purchase decisions of Web users. The nature of the reviewed items is extremely diverse, spanning everything from commercial products to restaurants and holiday destinations. As review-hosting websites become more popular, the number of available reviews per item increases dramatically. Even though this can be viewed as a healthy symptom of online information sharing, it can also be problematic for the interested user: as of February of 2010, Amazon.com hosted over 11,480 reviews on the popular “Kindle” reading device. Clearly, it is impractical for a user to read tsuch an overwhelming corpus in order to make a purchase decision. In addition, this massive volume of reviews inevitably leads to redundancy: many reviews are often repetitious, exhaustively expressing the same (or similar) opinions and contributing little additional knowledge. Further, reviews may also be misleading, reporting false information that does not accurately represent the attributes of an item. Possible causes of such reviews include:

- **Insufficient information:** The reviewer proceeds to an evaluation without having enough information on the item. Instead, opinions are based on partial or irrelevant information.
- **Fraud:** The reviewer maliciously submits false information on an item, in order to harm or boost its reputation.

The main motivation of our work is that a user should not have to manually go through massive volumes of redundant and ambiguous data in order to obtain the required information. The search engines that are currently employed by major review-hosting sites do not consider the particular nature of opinionated text. Instead, reviews are evaluated as typical text segments, while focused queries that ask for reviews with opinions on *specific* attributes are not supported. In addition, reviews are ranked based on very basic methods (e.g. by date) and information redundancy is not considered.

Ideally, false or redundant reviews could be filtered before they become available to users. However, simply labeling a review as “true” or “false” is over-simplifying, since a review may only be partially false. Instead, we propose a framework that evaluates the validity of the opinions expressed in a review and assigns an appropriate *confidence score*. High confidence scores are assigned to reviews expressing opinions that respect the *consensus* formed by the entire review corpus. For example, if 90% of the reviews compliment the battery-life of a new laptop, there is a strong positive consensus on the specific attribute. Therefore, any review that criticizes the battery-life will suffer a reduction in its confidence score, *proportional to the strength of the positive consensus*. At this point, it is important to distinguish between the two types of rare opinions: 1) those that are expressed on attributes that are rarely reviewed and 2) those that contradict the opinion of the majority of the reviewers on a specific attribute. Our approach only penalizes the latter, since the rare opinions in the first group can still be valid (e.g. expert opinions, commenting on attributes that are often overlooked by most users). Further, we employ a simple and efficient method to deal with ambiguous attributes, for which the numbers of positive and negative opinions differ marginally.

Confidence evaluation is merely the first phase of our framework; high-confidence reviews may still be redundant, if they express identical opinions on the same attributes. To address this, we propose an efficient redundancy filter, based on the skyline operator [55]. As shown in the experiments section, the filter achieves a significant reduction of the size of the corpus.

The final component of our framework deals with the evaluation of focused queries: given a set of attributes that the user is interested in, we want to identify *a minimal* set of high-confidence reviews that covers all the specified attributes. To address this, we formalize the *Review Selection* problem for large review corpora and propose a customized search engine for its solution. A complete diagram of our framework can be seen in Figure (4.1). Figure (4.2) shows a screenshot of **CREST** (Confident REview Search Tool), a user-friendly tool that implements the full functionality of our framework. In the shown example, **CREST** is applied on a corpus of reviews on a popular Las Vegas hotel. As soon as a review corpus is loaded, **CREST** evaluates the confidence of the available reviews and filters out redundant artifacts. The user can then select a set of features from a list extracted automatically from the corpus. The chosen set is submitted as a query to the search engine, which returns a compact and informative set of reviews. It is important to stress that our engine has no bias against attributes that appear sparsely in the corpus: as long as the user includes an attribute in the query, an appropriate review will be identified and included in the solution.

Contribution: Our primary contribution is an efficient search engine that is customized for large review corpora. The proposed framework can respond to any attribute-based query by returning an appropriate minimal subset of high-quality reviews.

Roadmap: We begin in Section 4.2 with a discussion on related work. In section 4.3 we introduce the Confident Search paradigm for large review corpora. In Section 4.4 we

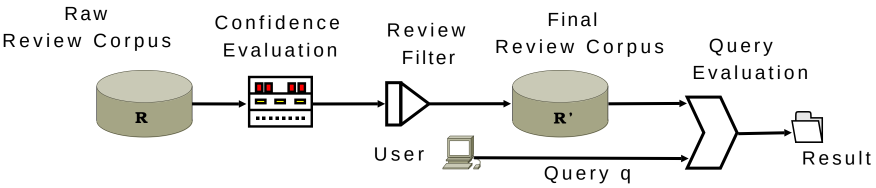
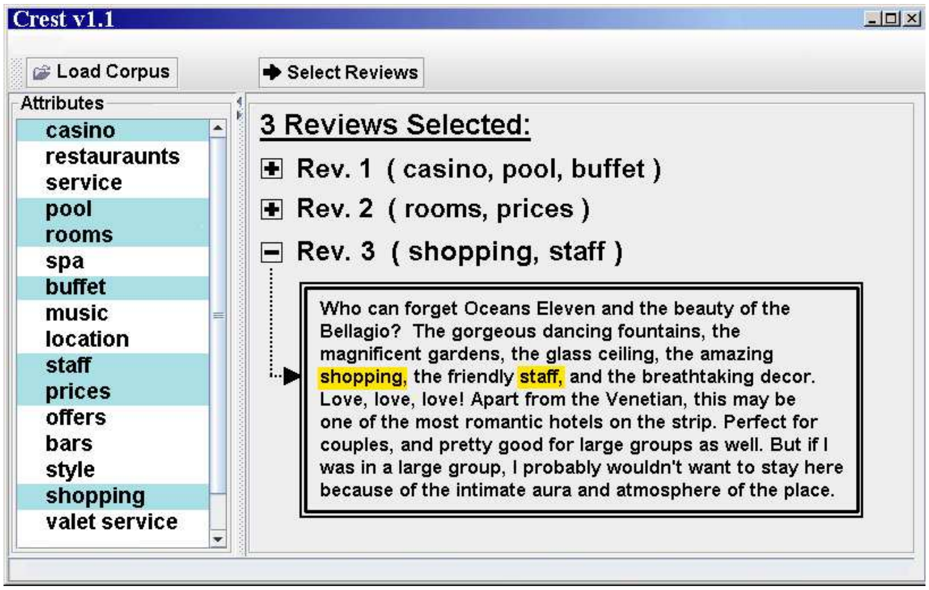


Figure 4.1: Given a review corpus $\mathcal{R}$, we first evaluate the *confidence* of each review $r \in \mathcal{R}$. Then, the corpus is filtered, in order to eliminate redundant reviews. Finally, given a query of attributes, the search engine goes through the processed corpus to evaluate the query and select an appropriate set of reviews.



3 Reviews Selected:

Rev. 1 (casino, pool, buffet)

Rev. 2 (rooms, prices)

Rev. 3 (shopping, staff)

Who can forget Oceans Eleven and the beauty of the Bellagio? The gorgeous dancing fountains, the magnificent gardens, the glass ceiling, the amazing shopping, the friendly staff, and the breathtaking decor. Love, love, love! Apart from the Venetian, this may be one of the most romantic hotels on the strip. Perfect for couples, and pretty good for large groups as well. But if I was in a large group, I probably wouldn't want to stay here because of the intimate aura and atmosphere of the place.

Figure 4.2: A user loads a corpus of reviews and then chooses a query of attributes from the automatically-extracted list on the left. The "Select Reviews" button prompts the system to return an appropriate minimal set of reviews.

describe how we measure the quality of a review through evaluating the confidence in the opinions it expresses. In Section 4.5 we discuss how we can effectively reduce the size of the corpus by filtering-out redundant reviews. In Section 4.6 we propose a review-selection mechanism for the evaluation of attribute-based queries. Then, in Section 4.7, we conduct a thorough experimental evaluation of the methods proposed in our work. Finally, we conclude in Section 4.8 with a brief discussion of the paper.

4.2 Background

Our work is the first to formalize and address the Confident Search paradigm for review corpora. Even though there has been progress in relevant areas individually, ours is the first work to synthesize elements from all of them toward a customized search engine for review corpora. Next, we review the relevant work from various fields.

Review Assessment: Some work has been devoted on the evaluation of review *helpfulness* [56, 57], formalizing the problem as one of regression. Jindal and Liu [58] also adopt an approach based on regression, focusing on the detection of spam (e.g. duplicate reviews). Finally, Liu and Cao [59] formulate the problem as binary classification, assigning a quality rating of “high” or “low” to reviews. Our concept of review assessment differs dramatically from the above-mentioned approaches: first, our framework has no requirement of tagged training data (e.g. spam/not spam, helpful/not helpful). Second, our work is the first to address redundant reviews in a principled and effective manner (Section 4.5). In any case, we consider prior work on review assessment complementary to ours, since it can be used to filter spam before the application of our framework.

Sentiment Analysis: Our work is relevant to the popular field of sentiment analysis, which deals with the extraction of knowledge from opinionated text. The domain of customer reviews is a characteristic example of such text, that has attracted much attention in the past [60, 61, 62, 63, 64, 65]. A particularly interesting area of this field is that of attribute and opinion mining, which we discuss next in more detail.

Attribute and Opinion Mining: Given a review corpus on an item, opinion mining [66, 67, 68, 69], looks for the attributes of the item that are discussed in each review, as well as the polarities (i.e. positive/negative) of the opinions expressed on each attribute. For our experiments, we implemented the technique proposed by Hu and Liu [66]: given a review corpus $\mathcal{R}$ on an item, the technique extracts the set of the item's attributes $\mathcal{A}$, and also identifies opinions of the form $(a \rightarrow p), p \in \{-1 + 1\}, \alpha \in \mathcal{A}$ in each review. We refer the reader to the original paper for further details. Even though this method worked superbly in practice, it is important to note that our framework is compatible with *any* method for attribute and opinion extraction.

Opinion Summarization: In the field of opinion summarization [59, 70, 71], the given review corpus is processed to produce a cumulative summary of the expressed opinions. The produced summaries are statistical in nature, offering information on the distribution of positive and negative opinions on the attributes of the reviewed item. We consider this work complementary to our own: we present an efficient search engine, able to select a minimal set of actual reviews in response to a specific query of attributes. This provides the user with actual comments written by humans, instead of a less user-friendly and intuitive statistical sheet.

4.3 Efficient Confident Search

Next, we formalize the *Confident Search* paradigm for large review corpora. We begin with an example, shown in Figure (4.3). The figure shows the attribute-set and the available review corpus $\mathcal{R}$ for a laptop computer. Out of the 9 available attributes, a user selects only those that interest him. In this case: {"Hard Drive", "Price", "Processor", "Memory"}. Given this query, our search engine goes through the corpus and selects a set of reviews $\mathcal{R}^* = \{r_1, r_7, r_9, r_{10}\}$ that accurately evaluates the specified attributes. Taking this example

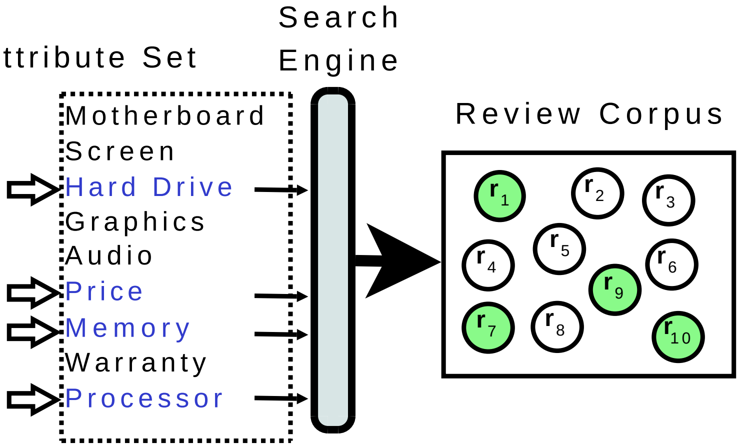


Figure 4.3: A use case of our search engine: The user submits a query of 4 attributes, selected from the attribute-set of a computer. Then, the engine goes through a corpus of reviews and locates those that best cover the query (highlighted circles).

into consideration, we can now define the three requirements that motivate our concept of *Confident Search*:

1. **Quality:** Given a query of attributes, a user should be presented with a set of high-quality reviews that accurately evaluates the attributes in the query.
2. **Efficiency:** The search engine should minimize the time required to evaluate a query, by appropriately pre-processing the corpus and eliminating redundancy.
3. **Compactness:** The set of retrieved reviews should be informative but also compact, so that a user can read through it in a reasonable amount of time.

Next, we will go over each of the three requirements, and discuss how they are addressed in our framework.

4.4 Quality through Confidence

We address the requirement for quality by introducing the concept of *confidence* in the opinions expressed within a review. Intuitively, a high-confidence review is one that provides accurate information on the item's attributes. Formally:

[Review Confidence Problem]: Given a review corpus $\mathcal{R}$ on an item, we want to define a function $conf(r, \mathcal{R})$ that maps each review $r \in \mathcal{R}$ to a score, representing the overall confidence in the *opinions* expressed within r.

Let $\mathcal{A}$ be the set of attributes of the reviewed item. Then, an *opinion* refers to one of the attributes in $\mathcal{A}$, and can be either positive or negative. Formally, we define an opinion as a mapping $(\alpha \rightarrow p)$ of an attribute $\alpha \in \mathcal{A}$ to a polarity $p \in \{-1, +1\}$. In our experiments, we extract the set of attributes $\mathcal{A}$ and the respective opinions using the method proposed in [66]. Further, let $\mathcal{O}^{-}_{r,a}$ and $\mathcal{O}^{+}_{r,a}$ represent the sets of negative and positive opinions expressed on an attribute α in review r, respectively. Then, we define $pol(\alpha, r)$ to return the *polarity* of α in r. Formally:

$$pol(\alpha, r) = \left\{ \begin{array}{l} +1, \text{ if } |\mathcal{O}^{+}_{r,\alpha}| > |\mathcal{O}^{-}_{r,\alpha}| \\ \\ -1, \text{ if } |\mathcal{O}^{+}_{r,\alpha}| < |\mathcal{O}^{-}_{r,\alpha}| \end{array} \right\} \tag{4.1}$$

Note that, for $|\mathcal{O}^{+}_{r,\alpha}| = |\mathcal{O}^{-}_{r,\alpha}|$, we simply ignore α, since the expressed opinion is clearly ambiguous. Now, given a review corpus $\mathcal{R}$ and an attribute α, let $n(\alpha \rightarrow p, \mathcal{R})$ be equal to the number of reviews in $\mathcal{R}$, for which $pol(\alpha, r) = p$. Formally:

$$n(\alpha \rightarrow p, \mathcal{R}) = |\{r : pol(\alpha, r) = p, \ r \in \mathcal{R}\}| \tag{4.2}$$

For example, if the item is a TV, then $n(\text{“screen”} \rightarrow +1, \mathcal{R})$ would return the number of reviews in $\mathcal{R}$ that express a positive opinion on its screen. Given Eq. (4.2), we can define the concept of the *consensus* of the review-corpus $\mathcal{R}$ on an attribute α as follows:

Definition 1. [Consensus]: Given a set of reviews $\mathcal{R}$ and an attribute α, we define the *consensus of* $\mathcal{R}$ *on* α as:

$$C_{\mathcal{R}}(a) = \operatorname*{argmax}_{p \in \{-1,+1\}} n(\alpha \rightarrow p, \mathcal{R}) \tag{4.3}$$

Conceptually, the consensus expresses the polarity $\in \{-1, +1\}$ that was assigned to the attribute by the majority of the reviews. Formally, given a review corpus $\mathcal{R}$ and an opinion $\alpha \rightarrow p$, we define the *strength* $d(\alpha \rightarrow p, \mathcal{R})$ of the opinion as follows:

$$d(\alpha \rightarrow p, \mathcal{R}) = n(\alpha \rightarrow p) - n(\alpha \rightarrow -p) \tag{4.4}$$

Since the consensus expresses the majority, we know that $d(\alpha \rightarrow C_{\mathcal{R}}(\alpha), \mathcal{R}) \geq 0$. Further, the higher the value of $d(\alpha \rightarrow C_{\mathcal{R}}(\alpha))$, the higher is our confidence in the consensus. Given Eq. (4.4), we can now define the overall confidence in the opinions expressed within a given review. Formally:

Definition 2. [Review Confidence]: Given a review corpus $\mathcal{R}$ on an item and the set of the item's attributes $\mathcal{A}$, let $\mathcal{A}_r \subseteq \mathcal{A}$ be the subset of attributes that are actually evaluated within a review $r \in \mathcal{R}$. Then, we define the overall *confidence* of r as follows:

$$conf(r, \mathcal{R}) = \frac{\sum_{\alpha \in \mathcal{A}_r} d(\alpha \rightarrow pol(\alpha, r), \mathcal{R})}{\sum_{\alpha \in \mathcal{A}_r} d(\alpha \rightarrow C_{\mathcal{R}}(\alpha), \mathcal{R})} \tag{4.5}$$

The confidence in a review takes values in $[-1, 1]$, and is maximized when all the opinions expressed in the review agree with the consensus (i.e. $pol(\alpha, r) = C_{\mathcal{R}}(\alpha), \forall \alpha \in \mathcal{A}_r$). By dividing by the sum of the confidence values in the consensus on each $\alpha \in \mathcal{A}_r$, we ensure that the effect of an opinion $(\alpha \rightarrow p)$ on the confidence of r is proportional to the strength of the consensus on attribute α.

High-confidence reviews are more trustworthy and preferable sources of information, while those with low confidence values contradict the majority of the corpus. The confidence scores are calculated offline and are then stored and readily available for the search engine to use on demand.

4.5 Efficiency through Filtering

In this Section, we formalize the concept of *redundancy* within a set of reviews and propose a filter for its elimination. As we show with experiments on real datasets, the filter can drastically reduce the size of the corpus. The method is based on the following observation:

Observation 1. Given two reviews r_1 and r_2 in a corpus $\mathcal{R}$, let $\mathcal{A}_{r1} \subseteq \mathcal{A}_{r2}$ and $pol(\alpha, r_1) = pol(\alpha, r_2), \forall \alpha \in \mathcal{A}_{r1}$. Further, let $conf(r_1, \mathcal{R}) \leq conf(r_2, \mathcal{R})$. Then r_1 is redundant, since r_2 expresses the same opinions on the same attributes, while having a higher confidence score.

According to Observation 1, some of the reviews in the corpus can be safely pruned, since they are *dominated* by another review. This formulation matches the definition of the well-known *Skyline* operator [55][72][73], formally defined as follows:

Definition 3. [Skyline]: Given a set of multi-dimensional points $\mathcal{K}$, $Skyline(\mathcal{K})$ is a subset of $\mathcal{K}$ such that, for every point $k \in Skyline(\mathcal{K})$, there exists no point $k' \in \mathcal{K}$ that *dominates* k. We say that k' *dominates* k, if k' is no worse than k in all dimensions.

The computation of the skyline is a highly-studied problem, that comes up in different domains [72]. In the context of our problem, the set of dimensions is represented by the set of possible opinions $\mathcal{O}_{\mathcal{R}}$ that can be expressed within a review corpus $\mathcal{R}$. In the general skyline scenario, a point can assume any value in any of its multiple dimensions. In our case, however, the value of a review $r \in \mathcal{R}$ with respect to an opinion *op* $\in \mathcal{O}_{\mathcal{R}}$ can only assume one of two distinct values: if the opinion is actually expressed in r, then the value on the respective dimension is equal to $conf(r, \mathcal{R})$. Otherwise, we assign a value of -1, which is the minimum possible confidence score for a review. This ensures that a review r_1 can never be dominated by another review r_2, as long as it expresses at least one opinion that is not expressed in r_2 (since the value of r_2 for the respective dimension will be the lowest possible, i.e. -1).

Most skyline algorithms employ multi-dimensional indexes and techniques for high-dimensional search. However, in a constrained space such as ours, such methods lose their advantage. Instead, we propose a simple and efficient approach that is customized for our problem. The proposed method, which we refer to as `ReviewSkyline`, is shown in Algorithm (2).

Algorithm 2 `ReviewSkyline`

```
    Input: review corpus $\mathcal{R}$, $conf(r, \mathcal{R}) \forall r \in \mathcal{R}$, set of possible opinions $\mathcal{O}_{\mathcal{R}}$
    Output: $Skyline$ of $\mathcal{R}$
 1: Sort all reviews in $\mathcal{R}$ in descending order by $conf(r, \mathcal{R})$
 2: Create an Inverted Index, mapping each opinion $op \in \mathcal{O}_{\mathcal{R}}$ to a list $L[op]$ of the reviews that
    express it, sorted by confidence.
 3: for every review $r \in \mathcal{R}$ do
 4:     if ($r$ is dominated by some set in $Skyline$) then
 5:         GOTO 3: // skip $r$
 6:     $\mathcal{L} = \{L[op] \mid \forall o \in \mathcal{O}_r\}$
 7:     while (NOT all Lists in $\mathcal{L}$ are exhausted) do
 8:         for every opinion $op \in \mathcal{O}_r$ do
 9:             $r' = getNext(L[op])$
10:             if $(conf(r, \mathcal{R}) < conf(r', \mathcal{R}))$ then
11:                 Consider $L[op]$ to be exhausted
12:                 GOTO 8:
13:             if ($r'$ dominates $r$) then
14:                 GOTO 3: // skip $r$
15:     $Skyline \leftarrow Skyline \cup \{r\}$
16: return $Skyline$
```

Analysis of Algorithm (2): The input consists of a review corpus $\mathcal{R}$, along with the confidence score of each review $r \in \mathcal{R}$ and the set of possible opinions $O_{\mathcal{R}}$. The output is the skyline of $\mathcal{R}$.

Lines [1-2]: The algorithm first sorts the reviews in descending order by confidence. This requires $O(|\mathcal{R}| \log |\mathcal{R}|)$ time. It then builds an inverted index, mapping each opinion to the list of reviews that express it, sorted by confidence. Since we already have a sorted list of all the review from the previous step, this can be done in $O(|\mathcal{R}| \times M)$ time, where M is the size of the review with the most opinions in $\mathcal{R}$.

Lines [3-15]: The algorithm iterates over the reviews in $\mathcal{R}$ in sorted order, eliminating reviews that are dominated by the current Skyline. In order to efficiently check for this, we keep he reviews in the Skyline sorted by confidence. Therefore, since a review can only be dominated by one of higher or equal confidence, a binary search probe is used to check if a review r is dominated.

In line (6), we define a collection of lists $\mathcal{L} = \{L[\textit{op}] | \forall \textit{op} \in \mathcal{O}_r\}$, where $L[\textit{op}]$ is the sorted list of reviews that express the opinion *op* (from the inverted index created in line (2)). The lists in $\mathcal{L}$ are searched in a round-robin fashion: the first $|\mathcal{L}|$ reviews to be checked are those that are ranked first in each of the lists. We then check the reviews ranked 2nd and continue until all the lists have been exhausted.

The $getNext(L[\textit{op}])$ routine returns the next review r' to be checked from the given list. If r' has a lower confidence than r, then we can safely stop checking $L[\textit{op}]$, since any sets ranked lower will have an even lower score. Therefore, $L[\textit{op}]$ is considered exhausted and we go back to check the list of the next opinion. If r' dominates r, we eliminate r and go back to examine the next review. If all the lists in $\mathcal{L}$ are exhausted without finding any review that dominates r, then we add it to the skyline.

Performance: In the worst case, all the reviews represent skyline points. Then, the complexity of the algorithm is quadratic in the number of reviews. In practice, however, the skyline includes only a small subset of the corpus. We demonstrate this on real datasets in the experiments section. We also show that `ReviewSkyline` is several times faster and more scalable than the state-of-the art for the general skyline computation problem. In addition, by using an inverted index instead of the multi-dimensional index typically employed by skyline algorithms, `ReviewSkyline` saves both memory and computational time.

4.6 Compactness through Selection

The requirement for compactness implies that simply evaluating the quality of the available reviews is not enough: top-ranked reviews may still express identical opinions on the same attributes and, thus, a user may have to read through a large number of reviews in order to obtain all the required information. Instead, given a query of attributes, a review should be included in the result, *only if it evaluates at least one attribute that is not evaluated in any of the other included reviews.* Note that our problem differs significantly from conventional document retrieval tasks: instead of independently evaluating documents with respect to a given query, we want a *set* of reviews that collectively cover a subset of item-features. In addition, we want the returned set to contain opinions that respect the consensus reached by the reviewers on the specified features. Taking this into consideration, we define the *Review Selection Problem* as follows:

Problem 1. [Review Selection Problem]: Given the review corpus $\mathcal{R}$ on an item and a subset of the item's attributes $\mathcal{A}^* \subseteq \mathcal{A}$, find a subset $\mathcal{R}^*$ of $\mathcal{R}$, such that:

1. All the attributes in $\mathcal{A}^*$ are covered in $\mathcal{R}^*$
2. $pol(\alpha, r) = C_{\mathcal{R}}(\alpha), \forall \alpha \in \mathcal{A}^*, r \in \mathcal{R}^*$.
3. Let $\mathcal{X} \subseteq 2^{\mathcal{R}}$ be the collection of review-subsets that satisfy the first 2 conditions. Then:
   $\mathcal{R}^* = \underset{\mathcal{R}' \in \mathcal{X}}{\operatorname{argmax}} \sum_{r \in \mathcal{R}'} conf(r, \mathcal{R}')$

The 1st condition is straightforward. The 2nd condition ensures that the selected reviews contain no opinions that contradict the consensus on the specified attributes, in order to avoid selecting reviews with contradictory opinions. Finally, the 3rd condition asks for the set with the maximum overall confidence, among those that satisfy the first 2 conditions.

Ambiguous attributes: For certain attributes, the number of negative opinions may be only marginally higher than the number of positive ones (or vice versa), leading to a weak consensus. In order to identify such attributes, we define the $weight$ of an attribute α to be proportional to the $strength$ of its respective consensus (defined in Eq. (4.4)). Formally, given a review corpus $\mathcal{R}$ and an attribute α, we define $w(\alpha, \mathcal{R})$ as follows:

$$w(\alpha, \mathcal{R}) = \frac{d(\alpha \rightarrow C_{\mathcal{R}}(\alpha), \mathcal{R})}{|\mathcal{R}|} \tag{4.6}$$

Observe that, since $0 \leq d(\alpha \rightarrow C_{\mathcal{R}}(\alpha) \leq |\mathcal{R}|$, we know that $w(\alpha, \mathcal{R})$ takes values in $[0, 1]$. Conceptually, a low weight shows that the reviews on the specific attribute are mixed. Therefore, a set of reviews that contains only positive (or negative) opinions will not deliver a complete picture to the user. To address this, we relax the 2nd condition as follows: if the weight of an attribute α is less than some pre-defined lower bound b (i.e. $w(\alpha, \mathcal{R}) < b$), then the reported set $\mathcal{R}^*$ will be allowed to include reviews that contradict the (weak) consensus on α. In addition, $\mathcal{R}^*$ will be $required$ to contain at least one positive and one negative review with respect to α. The value of b depends on our concept of a weak consensus. For our experiments, we used $b = 0.5$.

4.6.1 A Combinatorial Solution

Next, we propose a combinatorial solution for the Review Selection problem. We show that the problem can be mapped to the popular Weighted Set Cover problem [74, 75] (`WSC`), from which we can leverage solution techniques. Formally, the `WSC` problem is defined as follows:

[Weighted Set Cover Problem]: We are given a universe of elements $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$ and a collection $\mathcal{S}$ of subsets of $\mathcal{U}$, where each subset $s \in \mathcal{S}$ has a positive cost $cost[s]$. The problem asks for a collection of subsets $\mathcal{S}^* \subseteq \mathcal{S}$, such that $\bigcup_{s \in \mathcal{S}^*} \{s\} = \mathcal{U}$ and the cost $\sum_{s \in \mathcal{S}^*} cost[s]$ is minimized.

Given a review corpus $\mathcal{R}$, Routine (3) is used to generate a collection of sets $\mathcal{S}$, including a set s for every review $r \in \mathcal{R}$. The produced sets consist of elements from the same universe and have their respective costs, as required by the `WSC` problem.

Routine 3 `Transformation Routine`

```
Input: Set of attributes $\mathcal{A}$, Set of reviews $\mathcal{R}$
Output: Collection of subsets $\mathcal{S}$, $cost[s] \forall s \in \mathcal{S}$
for (every review $r \in \mathcal{R}$) do
    $s \leftarrow \emptyset$ // New empty set
    for (every attribute $\alpha \in \mathcal{A}$) do
        if $pol(\alpha, r) = +1$ then $s \leftarrow s \cup \{\alpha^+\}$
        else if $pol(\alpha, r) = -1$ then $s \leftarrow s \cup \{\alpha^-\}$
    $cost[s] \leftarrow (1 - conf(r, \mathcal{R}))/2$
    $\mathcal{S}.add(s)$
return $\mathcal{S}, cost[\,]$
```

The `Greedy-Reviewer` Algorithm: Next, we present an algorithm that can efficiently solve the Review Selection problem. The input consists of the collection of sets $\mathcal{S}$ returned by the transformation routine, a query of attributes $\mathcal{A}^* \subseteq \mathcal{A}$, and a number $b \in [0, 1]$, used to determine if the consensus on an attribute is weak (as described earlier in this section). The algorithm returns a subset $\mathcal{S}^*$ of $\mathcal{S}$. The pseudocode is given in Algorithm (1).

The Algorithm begins by populating the universe $\mathcal{U}$ of elements to be covered (lines 2-6). For each attribute $\alpha \in \mathcal{A}^*$, if the consensus on the attribute is weak ($w(\alpha, \mathcal{R}) < b$), two elements α^+ and α^- are added to $\mathcal{U}$. Otherwise, if the consensus is strong and positive (negative), an element α^+ (α^-) is added.

The universe of elements $\mathcal{U}$, together with the collection of sets $\mathcal{S}$, constitute an instance of the `WSC` Problem. The problem is known to be NP-Hard, but can be approximated by a well-known Greedy algorithm, with an $\ln n$ approximation ratio [75]. First, we define 2 variables $\mathcal{S}^*$ and $\mathcal{Z}$ to maintain the final solution and the still-uncovered subset of $\mathcal{U}$, respectively. The greedy-choice is conducted in lines 9-11: the algorithm selects the set that minimizes the quotient of the cost, over the still-uncovered part of $\mathcal{U}$ that is covered by the set. Since there is a 1-to-1 correspondence between sets and reviews, we can trivially

Algorithm 1 `Greedy-Reviewer`

```
Input: $\mathcal{S}, \mathcal{A}^* \subseteq \mathcal{A}$, lower bound $b$
Output: weighted set-cover $\mathcal{S}^*$
$\mathcal{U} \leftarrow \emptyset$
for every attribute $\alpha \in \mathcal{A}^*$ do
    if $w(\alpha, \mathcal{R}) < b$ then $\mathcal{U} \leftarrow \mathcal{U} \cup \{\alpha^+\} \cup \{\alpha^-\}$
    else if $C_{\mathcal{R}}(\alpha) = +1$ then $\mathcal{U} \leftarrow \mathcal{U} \cup \{\alpha^+\}$
    else $\mathcal{U} \leftarrow \mathcal{U} \cup \{\alpha^-\}$
$\mathcal{S}^* \leftarrow \emptyset$ // The set-cover
$\mathcal{Z} \leftarrow \emptyset$ // The still-uncovered part of $\mathcal{U}$
while ($\mathcal{S}^*$ is not a cover of $\mathcal{U}$) do
    $s \leftarrow \underset{s' \in \mathcal{S},\, s' \cap \mathcal{U} = \emptyset}{\operatorname{argmin}} \left( \frac{cost[s']}{|s' \cap \mathcal{Z}|} \right)$
    $\mathcal{S}^*.add(s)$
return $\mathcal{S}^*$
```

obtain the set of selected reviews $\mathcal{R}^*$ from the reported set-cover $\mathcal{S}^*$ and return it to the user.

4.7 Experiments

In this section, we present the experiments we conducted toward the evaluation of our search framework. We begin with a description of the used datasets. We then proceed to discuss the motivation and setup of each experiment, followed by a discussion of the results. All experiments were run on a desktop with a Dual-Core 2.53GHz Processor and 2G of RAM.

4.7.1 Datasets

• `GPS`: For this dataset, we collected the complete review corpora for 20 popular GPS Systems from Amazon.com. The average number of reviews per item was 203.5. For each review, we extracted the stars rating, the date the review was submitted and the review content.

- TVs: For this dataset, we collected the complete review corpora for 20 popular TV Sets from Amazon.com. The average number of reviews per item was 145. For each review, we extracted the same information as in the GPS dataset.

- Vegas-Hotels: For this dataset, we collected the review corpora for 20 popular Las Vegas Hotels from yelp.com. Yelp is a popular review-hosting website, where users can evaluate business and service providers from different parts of the United States. The average number of reviews per item was 266. For each review, we extracted the content, the stars rating and the date of submission.

- SF-Restaurants: For this dataset, we collected the reviews for 20 popular San Francisco restaurants from yelp.com. The average number of reviews per item was 968. For each review, we extracted the same information as in the Vegas-Hotels dataset. **The data is available upon request.**

4.7.2 Qualitative Evidence

We begin with some qualitative results, obtained by using the proposed search framework on real data. For lack of space, we cannot present the sets of reviews reported for numerous queries. Instead, we focus on 2 indicative queries, 1 from SF-Restaurants and 1 from Vegas-Hotels. For reasons of discretion, we omit the names of the specific items. For each item, we present the query, as well as the relevant parts of the retrieved reviews.

SF-Restaurants

Item 1, Query: {*food, service, atmosphere, restrooms*} 3 Reviews:

- “*...The dishes were creative and delicious ... The only drawback was the single unisex restroom.*”

- “*Excellent food, excellent service. Only taking one star for the size and cramp seating. The wait can get long, and i mean long...*”

- “*... Every single dish is amazing. Solid food, nice cozy atmosphere, extremely helpful waitstaff, and close proximity to MY house...*”

Item 2, Query: {*location, price, music*}, 2 Reviews:

- *“...Great location, its across from 111 Minna. Considering the decor, the prices are really reasonable...”*

- *“..Another annoying thing is the noise level. The music is so loud that it’s really difficult to have a conversation...”*

`Vegas-Hotels`

Item 3, Query: {*pool, location, rooms*}, 1 Review:

- *“...It was also a fantastic location, right in the heart of things...The pool was a blast with the eiffel tower overlooking it with great frozen drinks and pool side snacks. The room itself was perfectly fine, no complaints.”*

Item 4, Query: {*pool, location, buffet, staff* }, 2 Reviews:

- *“This is one of my favorite casinos on the strip; good location; good buffet; nice rooms; nice pool(s); huge casino...”*

- *“...The casino is huge and there is an indoor nightclub on the ground floor. All staff are professional and courteous...”*

As can be seen from the results, our engine returns a compact set of reviews that accurately captures the consensus on the query-attributes and, thus, serves as a valuable tool for the interested user.

4.7.3 Skyline Pruning for Redundant Reviews

In this section, we present a series of experiments for the evaluation of the redundancy filter described in Section 4.5.

Number of Pruned Reviews: First, we examine the percentage of reviews that are discarded by our filter: for every item in each of the 4 datasets, we find the set of reviews that represents the skyline of the item’s review corpus. We then calculate the average percentage of pruned reviews (i.e. reviews not included in the skyline), taken over all the items in each dataset. The computed values for `TVs`, `GPS`, `Vegas-Hotels` and `SF-Restaurants` were $0.4, 0.47, 0.54$ and 0.79, respectively. The percentage of pruned reviews reaches up to

79%. This illustrates the redundancy in the corpora, with numerous reviewers expressing identical opinions on the same attributes. By focusing on the skyline, we can drastically reduce the number of reviews and effectively reduce the query response time.

Evolution of the Skyline: Next, we explore the correlation between the size of the skyline and the size of the review corpus, as the latter grows over time. First, we sort the reviews for each item in ascending order, by date of submission. Then, we calculate the cardinality of the skyline of the first K reviews. We repeat the process for $K \in \{50, 100, 200, 400\}$. For each value of K, we report the average percentage of the reviews that is covered by the skyline, taken over all the items in each dataset. The results are shown in Table 4.1.

Table 4.1: Skyline Cardinality Vs. Total #Reviews

	Avg #Reviews in the Skyline (Per Item)			
#Reviews	TVs	GPS	Vegas-Hotels	SF-Restaurants
50	0.64	0.53	0.47	0.35
100	0.56	0.47	0.44	0.28
200	0.55	0.43	0.4	0.24
400	0.55	0.43	0.39	0.19

The table shows that the introduction of more reviews has a decreasing effect on the percentage of the corpus that is covered by the skyline, which converges after a certain point. This is an encouraging finding, indicating that a compact skyline can be extracted regardless of the size of the corpus.

Running Time: Next, we evaluate the performance of the ReviewSkyline algorithm (Section 4.5). We compare the required computational time against that of the state-of-the-art Branch-and-Bound Algorithm (BnB) by Papadias et al. [72]. Our motivation is to show how our specialized algorithm compares to one made for the general problem.

The results, shown in Table 4.2, show that ReviewSkyline achieved superior performance in all 4 datasets. BnB treats each corpus as a very-high dimensional dataset, assuming a new dimension for every distinct opinion. As a result, the computational time is dominated by the construction of the required R-tree structure, which is known to deteriorate for very high dimensions [76]. ReviewSkyline avoids these shortcomings by taking into consideration the constrained nature of the review space.

Table 4.2: Avg Running Time Skyline Computation (in seconds)

	TVs	GPS	Vegas-Hotels	SF-Restaurants
ReviewSkyline	0.2	0.072	0.3	0.11
BnB	24.8	39.4	28.9	116.2

Scalability: In order to demonstrate the scalability of ReviewSkyline, we created a benchmark with very large batches of artificial reviews. As a seed, we used the reviews corpus for the “slanted door” restaurant from the SF-Restaurants dataset, since it had the largest corpus across all datasets (about 1400 reviews). The data was generated as follows: first, we extracted the set $\mathcal{Y}$ of distinct opinions (i.e. attribute-to-polarity mappings) from the corpus, along with their respective frequencies. A total of 25 distinct attributes were extracted from the corpus, giving us a set of 50 distinct opinions. In the context of the skyline problem, this number represents the dimensionality of the data.

Each artificial review was then generated as follows: first, we flip an unbiased coin. If the coin comes up heads, we choose an opinion from $\mathcal{Y}$ and add it to the review. The probability of choosing an opinion from $\mathcal{Y}$ is proportional to its frequency in the original corpus. We flip the coin 10 times. Since the coin is unbiased, the expected average number of opinions per review of is 5, which is equal to the actual average observed in the corpus. We created 6 artificial corpora, where each corpus had a population of p reviews, $p \in \{10^4, 2 \times 10^4, 4 \times 10^4, 8 \times 10^4, 16 \times 10^4\}$. We compare ReviewSkyline with the BnB Algorithm, as we did in the previous experiment. The Results are shown in Figure (4.4). The entries on the x-axis represent the 5 artificial corpora, while the values on the y-axis represent the computational time *(in logarithmic scale)*. The results show that ReviewSkyline achieves superior performance for all 5 corpora. The algorithm exhibited great scalability, achieving a low computational time even for the largest corpus (less than 3 minutes). In contrast to ReviewSkyline, BnB is burdened by the construction and poor performance of the R-tree in very high-dimensional datasets.

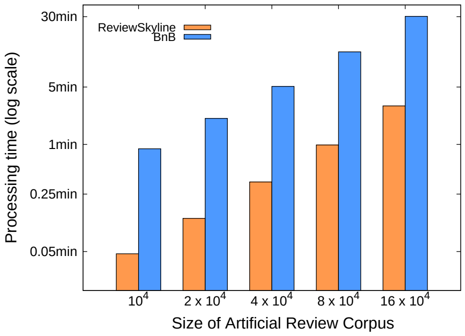
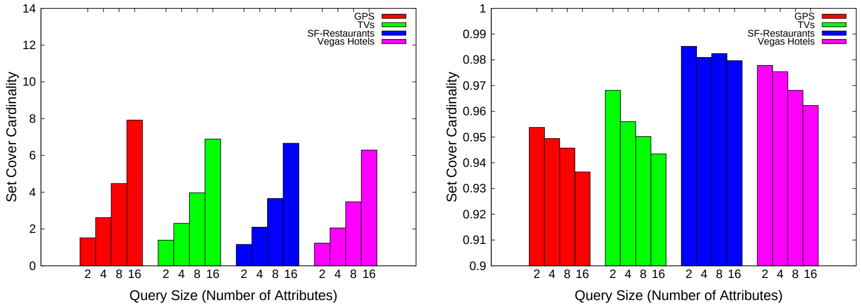


Figure 4.4: Scalability of `ReviewSkyline` and `BnB`

4.7.4 Query Evaluation

In this section, we evaluate the search engine described in Section 4.6. Given the set of attributes $\mathcal{A}$ of an item, we choose 100 subsets of $\mathcal{A}$, where each subset contains exactly k elements. The probability of including an attribute to a query is proportional to the attribute's frequency in the corpus. The motivation is to generate more realistic queries, since users tend to focus on the primary and more popular attributes of an item. We repeat the process for $k \in \{2, 4, 8, 16\}$, for a total of $100 \times 4 = 400$ queries per item.

Query size Vs. Result size: First, we evaluate how the size of the query affects the cardinality of the returned sets. Ideally, we would like to retrieve a small number of reviews, so that a user can read them promptly and obtain the required information. Given a specific item I and a query size k, let $Avg[I, k]$ be the average number of reviews included in the result, taken over the 100 queries of size k for the item. We then report the mean of the $Avg[I, k]$ values, taken over all 20 items in each dataset. The results are shown in Figure 4.5(a): The reported sets were consistently small, with less than 8 reviews were enough to cover queries containing up to 16 different attributes. Such compact sets are desirable since they can promptly be read by the user.



(a) Avg. Result size

(b) Avg. Confidence per review

Figure 4.5: Figures (a) and (b) show the average number of reviews included in the result and the average confidence per reported review, respectively.

Query Size Vs. Confidence: Next, we evaluate how the size of the query affects the average confidence of the selected reviews. The experimental setup is similar to that of the previous experiment. However, instead of the average result cardinality, we report the average confidence per selected review. Figure 4.5(b) shows the very promising results. An average confidence of 0.93 or higher was consistently reported for all query sizes, and for all 4 datasets. Combined with the findings of the previous experiment, we conclude that our framework produces compact sets of high-quality reviews.

4.8 Conclusion

In this work, we formalized the Confident Search paradigm for large review corpora. Taking into consideration the requirements of the paradigm, we presented a complete search framework, able to efficiently handle large sets of reviews. Our framework employs a principled method for evaluating the confidence in the opinions expressed in reviews. In addition, it is equipped with an efficient method for filtering redundancy. The filtered corpus maintains all the useful information and is considerably smaller, which makes it easier to store and to search. Finally, we formalized and addressed the problem of selecting a minimal set of high-quality reviews that can effectively cover any query of attributes submitted by the user. The efficacy of our methods was demonstrated through a rigorous and diverse experimental evaluation.

Acknowledgements: This research was partially supported by the MODAP and the SemsorGrid4Env EU projects and by NSF.

Chapter 5

Ranking by Comprehensibility in Foreign Document Retrieval

5.1 Introduction

Large numbers of texts discussing the same topic can nowadays be retrieved from Web sources around the world (e.g. news portals, reviews, blogs, RSS feeds, etc.). As a result, a typical web search may return similar documents in multiple languages. The question that we are addressing in this work is how to build an engine that delivers not only the most relevant documents, but also the ones that best match the user's comprehension level of a foreign language. Foreign documents that are easier to read and understand should be ranked higher than more advanced texts with the same coverage of the topic.

This research direction is becoming increasingly prominent, due to the wide accessibility of worldwide text and information resources. Since recently, the Google Search engine allows user to focus their search on webpages from one of three “Reading Levels” (Basic, Intermediate, Advanced). Even though the filter is only applicable to English texts, the motivation is similar to our own. A core operation that we offer in this work is an automated methodology for *sorting* foreign documents based on their easiness of understanding. Therefore, given a collection of foreign documents (whether these are books, articles, or simply short news events), we provide a structured methodology to effectively and accurately rank them based on their estimated *comprehensibility*. In this work, we embed this mechanism into an online search paradigm that not only retrieves relevant documents, but also ranks them based on their perceived difficulty. To the best of our knowledge, this is the first approach that examines the problem of document ranking through the prism of foreign language difficulty, using a completely unsupervised approach.

The problem is challenging because it lies at the confluence of fields as diverse as linguistics, information retrieval and machine learning. Our approach combines both structural and linguistic features, exploring the different aspects of document comprehensibility. In order to evaluate the difficulty of each foreign word, we leverage the knowledge distilled from large corpora of web documents. This enables us to infer the difficulty of individual terms and, consequently, of a document as a whole.

An additional dimension that we consider when estimating the reading difficulty of a foreign document, is the native language of the reader. For example, for a native Portuguese speaker, it can be significantly easier to comprehend Spanish documents rather than documents written in Greek or German. This is mainly due to the presence of *cognates*, i.e., words that are similar in both meaning and form in two languages. Such visual similarities between words can significantly ease the task of a reader. We present techniques that identify such word instances and adjust the perceived document difficulty accordingly.

With the proliferation, digitization and availability of increasingly larger text corpora (e.g. through electronic bookstores) one can expect a surge of interest in the technology explicated in this work. We envision numerous applications where our methodology can be of use:

1) **Language-comprehension and personalization of the Web**. Using our approach one can rank and present ‘similar’ news articles to a foreign language user, based on the perceived comprehension of the article, i.e., from most basic to most advanced usage of

the foreign language. Augmented with an interactive phase that allows the user to input one's own linguistic skills (i.e., the level of his/her proficiency in different languages), we provide additional building blocks to a multilingual personalization of the web. In addition, such technology can be used to decide *when* to translate a foreign document, by leveraging the estimated document comprehension difficulty.

2) **Online Bookstores**. Imagine the case of an English speaking reader interested in German literature books. Which one should he/she read based on one's reading and communication skills? The approach that we present in this work presents a direct solution to this problem.

3) **Education.** Many studies have suggested that learning a foreign language is more effective when studying texts match one's comprehension level [77]. Therefore, the tools provided in our work can be used to recommend the most suitable reading material to foreign language students.

In addition to bringing to the foreground the practical problem of how to rank foreign language documents based on their comprehension difficulty, our work makes various technical contributions: i) We present a sound methodology for evaluating the comprehensibility of foreign documents given the user's native language. We provide methodologies for extracting the required textual features and put forward distance measures to evaluate difficulty. Our methods are language-independent and have no requirement for training data that are pre-tagged for comprehensibility purposes. ii) We describe a skyline-based ranking approach in order to guide the user through documents of varying relevance and comprehension. iii) Finally, by using the German language as the core of our analysis, we provide evidence of the applicability of our techniques, when addressed to English speaking users.

5.2 Overview

Given a query and a set of documents, the goal is to retrieve results that are both relevant to the query and understandable by the user. Thus, we identify *relevance* and *comprehensibility* as the two main factors in our setting.

Relevance: Relevance search of a query toward a set of documents is nowadays a well-studied research topic, with many mature solutions already offered by the diverse search engines. Popular metrics of relevance include the cosine similarity measure [78], latent semantic models [79], or more advanced graph-based techniques such as Google's Pagerank [80].It is also common for document relevance models to incorporate the parameter of *diversity* [81], so as to offer better topic coverage in the least amount of presented results.

In this work we will not elaborate further on how to measure relevance, which we assume as given by the application. Our focus is on the comprehensibility component.

Comprehensibility: To estimate the comprehensibility of a foreign document we consider aspects such as:

a) The *structural difficulty* of the text, that is, how lengthy or perplexed the structure is.

b) The perceived *vocabulary difficulty*, which can be estimated by examining both how 'popular' a word is (e.g. a frequently encountered term is more likely to be better understood), and how similar it looks to its translation in the user's native language. A word that has similar form and meaning in both languages (a *cognate*) is considered easier to understand.

Given the abundance of numerous Information Retrieval (IR) applications, which already provide the relevance part of the search, we seek to provide an abstraction layer for easily encapsulating a comprehensibility metric on top of any pre-existing relevance metric. The capability of meta-application for the foreign document comprehensibility metric is of significant practical importance. We achieve this with the aid of *skyline* operators.

By viewing relevance and comprehensibility as two (possibly conflicting) axis of search,

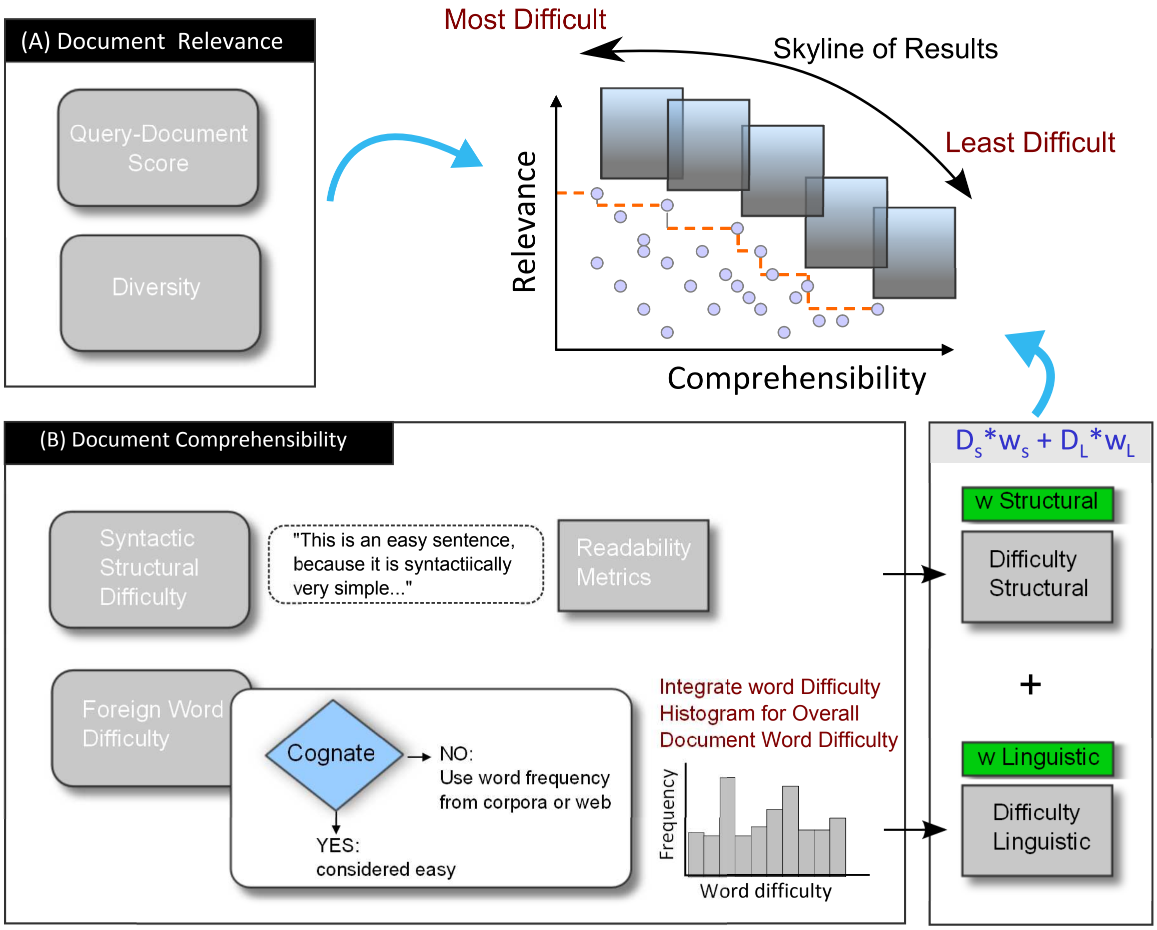


Figure 5.1: Overview of our approach for incorporating foreign document comprehensibility in Information Retrieval.

the *Pareto boundary* or *skyline* [55] of the two dimensions will include only the best documents in terms of the two examined parameters. Therefore, in our context, documents can be visualized as points in a 2-dimensional space, dictated by the comprehensibility and relevance values. Recall that, given a set of multidimensional points $\mathcal{X}$, a point is included in the skyline of $\mathcal{X}$ if it has a higher value in at least one dimension, compared to all the other points in the set. Alternatively, if a point is worse than another across all dimensions, we say that it is *dominated* and is thus excluded from the skyline. The set of documents on the skyline serves as a compact response to the user's query. In addition, such a 2D construction can easily accommodate visualization or user-preference scenarios ("find more relevant or more comprehensible documents").

An overview of our methodology as described above is summarized in Figure 5.1. In the remainder of the work we elaborate on how to estimate the document comprehensibility, essentially forming a *rank* operator of how understandable a document is by a non-native speaker. Finally, relevance and comprehensibility are fused together and the document-set skyline contains the most promising text candidates (top-right part of Figure 5.1).

Before, we delve into the technical considerations of our approach, we briefly revisit some previous relevant work and position our contributions accordingly.

5.3 Related Work

The field that we are examining is related to the problem of document *readability* [82]. However, the problem that we are focusing on is much richer and more elaborate, since one has to assess the difficulty of a *foreign* document. This depends not only on structural features of the sentence, but mostly on linguistic features, such as the number and the difficulty of the unknown foreign words.

Work on *text readability* can be broadly categorized, from a machine learning perspective, into supervised and unsupervised.

Unsupervised approaches mainly rely on two aspects of text: the familiarity of the reader with its semantic units (words or phrases) and the complexity of its syntax. In order to define a metric for the former, linguistic resources ranging from manually compiled lists of words [83] to language models [84] have been employed. In order to define a metric for syntactic complexity, the average sentence length is widely used, since it has been found to be strongly correlated with comprehensibility [85, 86]. The *Flesch Reading Ease* measure is often used as a baseline to compare against, when measuring the accuracy of supervised approaches [87]. In this work we utilize this metric as a weighted factor for partially estimating the syntactic difficulty of a foreign document.

Supervised approaches exploit the availability of training data in order to derive statistical language models of readability for a particular language. This type of approaches can be further divided into: *text categorization approaches* [88, 89], and *learn-to-rank approaches* [90]. In the former case, training data consists of categories associated by human subjects to documents, where the categories reflect the text difficulty and feature an order relationship. The learn-to-rank approaches address the problem as one of pairwise ranking of documents: rather than trying to predict the readability of a single document, the goal is to determine which document in a pair of documents is more difficult – doing that for every pair of documents.

Works that consider readability of foreign documents for educational purposes have been explored, among others, by Ott [91] and Uitdenbogerd [92]. These examine the possibility of combining already existing readability metrics into new through weighted average. Limitations of such work include the fact that no automatic recognition of cognates is considered.

The topic of language *cognates* and false friends (words that appear to be cognates but are not) has been studied in many linguistic experiments [93], particularly because it has been noted that translations between cognate words are easier to acquire when learning a new language. Efforts for automatic mapping between cognates have appeared in [94] for Spanish and Portuguese words, using subword dictionaries and thesauri in conjunction with substitution patterns. Finally, Mitkov et al. [95], presented automated techniques for identification of cognates and false friends through examination of bilingual corpora.

To the best of our knowledge, the only other work in the literature that considers the problem of readability of foreign documents while taking into account the user's native and near-native language(s), is the work by [96] on supervised readability prediction. Contrary to that work, however, our techniques do not depend on any sort of training data, making them more flexible and widely applicable. Additionally, our work is the first to explore a number of aspects of document comprehensibility, including automatic cognate recognition, and combines all of them toward an efficient mechanism for the ranking of foreign documents.

5.4 Comprehensibility

In this section, we describe the proposed mechanism for evaluating the comprehensibility of a foreign document. The two primary factors of our evaluation are *readability*, which assesses the structural features of a given document, and *familiarity*, which focuses on the vocabulary. Each of these two components captures a different aspect of comprehensibility. A sketch of our comprehensibility-evaluation mechanism is shown in Figure 5.2.

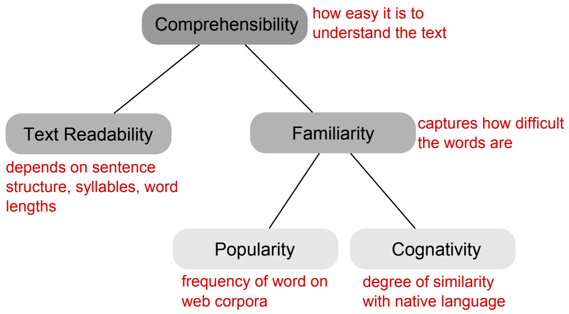

Comprehensibility
how easy it is to understand the text
Text Readability
depends on sentence structure, syllables, word lengths
Familiarity
captures how difficult the words are
Popularity
frequency of word on web corpora
Cognativity
degree of similarity with native language

Figure 5.2: The factors affecting document comprehensibility in our model

The comprehensibility of a document d with respect to a language L is defined as a linear combination of readability and familiarity. Formally:

$$C(d, L) = w_1 \times fam(d, L) + w_2 \times rd(d), \tag{5.1}$$

where $fam(d, L)$ denotes the familiarity of document d in language L, and $rd(d)$ denotes the readability of d, and where $w_1, w_2 \geq 0$ are the weights. Notice that familiarity (and hence comprehensibility) is defined as a function of the target language L. For example, a German document is expected to have higher comprehensibility value when read by Dutch people rather than by Italian people due to the smaller linguistic divergence of these two languages.

Finally, the two non-negative weights w_1 and w_2 provide the flexibility of tailoring comprehensibility to specific application criteria. For example, in cases when documents are known to consist of terms from the same constrained vocabulary (e.g. a collection of medical documents), it is reasonable to give higher priority to structural readability, rather than vocabulary-based measures. On the other hand, vocabulary-based measures have more merit in diverse collections (e.g. articles from different sources on the Web). Later, in Section 5.5 we propose an automated method to *learn* for parameter estimation given a training set of user rankings.

5.4.1 Text Readability

The notion of document readability has been a well-studied topic, particularly for English documents. One of the first essays on the topic was completed by Sherman on his 1893 study [97]. By analyzing various documents that spanned extended time periods, Sherman observed that the average sentence length was dropping with time: approximately 23 words per sentence at his time, down from 50 words at the Pre-Elizabethan area. Sherman was the first one to propose a purely statistical analysis of literature, and also made the important observation that shorter sentences tend to increase readability.

Since then, many readability formulas have been proposed, all attempting to assign a single numerical readability score to each document. Flesch, Kincaid and Zipf, all noted a mathematical relationship between the frequency of easy and difficult words, with the majority of the readability formulas inherently penalizing polysyllabic words and long, complex sentences.

A popular example of a readability formula is the Flesch Reading Ease (FRE) measure [98], which consists of a linear function of the mean number of syllables per word and the mean number of words per sentence in the document. This measure does not employ direct estimates of word frequency, but it instead relies on heuristic weights which attempt to capture the idea in the spirit of Zipf's Law [99] – that more frequent words are likely to have fewer syllables. Since our primary focus is on German documents, in our evaluations we employ an instance of the Flesch measure with its weights adapted to German documents [100]:

$$FRE(d) = 180 - \frac{words(d)}{sents(d)} - 58.5 \times \frac{syllables(d)}{words(d)}, \tag{5.2}$$

where $words(d), sents(d)$ and $syllables(d)$ denote the number of words, sentences and syllables in d, respectively. The weights on the above formula have been derived by means of regression on training data. The Flesch Reading Ease yields numbers from 0 to 100, expressing the range from ‘very difficult’ to ‘very easy’, and is meant to be used for measuring the readability of texts addressed to adult language users. Alternative readability measures include the Automated Readability Index (ARI), the Coleman-Liau Index, the SMOG Index and others. All these use the same building blocks as FRE in order to evaluate a given document. We choose FRE for its popularity and its use as a standard for readability by many organizations (e.g., by the U.S. Department of Defense).

Finally, we define the readability $rd(d)$ of a document d as the normalized version of $FRE(d)$, taken by dividing the score with the maximum $FRE(\cdot)$ observed over our entire document collection. More precisely, given a document collection $\mathcal{D}$, the readability of a document $d \in \mathcal{D}$ is defined by

$$rd(d) := \frac{FRE(d)}{\max_{d' \in \mathcal{D}} FRE(d')}, \tag{5.3}$$

taking values in $[0, 1]$.

5.4.2 Familiarity

The familiarity of a document assesses how likely it is that the vocabulary used is known to the user. We define the measure as a function of two indicators: *popularity* and *cognativity*.

Popularity is used to capture the frequency of the document terms in texts written in the language under consideration; intuitively, rare terms are less likely to be familiar to the user. Cognativity is a language dependent (or even user dependent) measure. Its use is to capture the degree to which a document's terms are similar in the user's own native language; normally, such terms would be easier to understand. Next, we discuss these two factors of comprehensibility in more detail.

Popularity

Let us define V_d to be the vocabulary of a given document d. This vocabulary consists of the distinct tokens present in d, including terms, n-grams and phrases. Populating V_d depends on the particular tokenizer one applies and features standard processing steps such as stemming or removal of stopwords. Apart from their frequency in d, these tokens also have a (prior) global frequency-based measure of *popularity*, indicating how commonly they appear in documents of a given language. When browsing through a foreign document, a non-native speaker is more likely to recognize a very popular token than one which is rarely used. In a broader context, a document consisting mostly of commonly used tokens is much easier to comprehend than another that uses more esoteric and unfamiliar vocabulary. In order to capture this "prior frequency" of a given a token t, we utilize the *collective knowledge* of the web.

Even though most previous work on text readability utilized large text corpora for estimating prior frequency of a token, in our approach we estimate the popularity of token through its instances on the web. Nowadays, most search engines provide the number of pages that the query appears in. We use this information as an estimate of term popularity [1]. An added advantage of using search engines instead of pre-existing text corpora, is the fact that online texts capture newly used terms, which is important since languages constitute an evolving organism. For example, the popularity of recent technical terms like Apple's

[1] Specifically, we use the page count from the Google search engine.

‘iPad’ can be easily deduced by its widespread presence on web texts. Finally, search engines provide the functionality of focusing on a particular *language* for the documents to be retrieved.

Formally, *popularity* is defined as:

Definition 6 (Popularity)**.** *The popularity of a term t is computed as the fraction:*

$$pop(t) = |\{t' : count(t') < count(t), t' \in \mathcal{V}\}|/|\mathcal{V}|, \tag{5.4}$$

where $count(t)$ returns the number of appearances of a given token t in the entire document collection $\mathcal{D}$, and $\mathcal{V}$ is the vocabulary of all the distinct tokens in $\mathcal{D}$. The popularity of t is thus defined as the percentage of tokens in $\mathcal{V}$ that have less appearances in $\mathcal{D}$ than t.

In addition to having a clear probabilistic interpretation, this formula is robust to outliers (i.e., tokens with very low or very high frequencies) and serves as an intuitive and parameter-free way to smooth the obtained counts. Alternative smoothing techniques have been proposed in the literature [101].

Cognativity

Cognates are words in related languages that exhibit orthographic and semantic affinity. They may originate from the same ancestor word, or they could simply be *loan* words (e.g., the word ‘computer’). As an illustrative example, the German noun ‘Haus’ corresponds to the English word ‘house’. Similarly the German adjective ‘politisch’ easily maps to ‘political’ in English.

Identifying cognates in a text is important, since they affect bilingual language processing; presence of large number of cognates in a text can enhance its comprehensibility. For example, a user proficient in English can easily deduce that the German sentence “Ein Experte kam die Maschine zu reparieren” translates to “An expert came to repair the machine”, even if one’s German skills are basic.

We employ a simple approach for spotting cognate words, by exploiting the interlingual homography. Our approach is based on the variation of the Longest Common Subsequence (LCSS). In particular, given a term t, let $\texttt{tr}(t, L)$ be its translation in the native language L of the user. Then, their similarity sim is defined as:

$$sim(t, \texttt{tr}(t, L)) = \frac{LCSS(t, \texttt{tr}(t, u))}{max(|t|, |\texttt{tr}(t, L)|)}$$

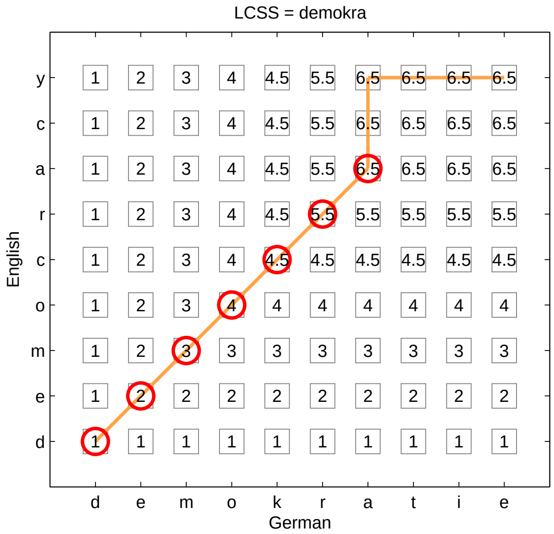
where $|\cdot|$ represents the length of a term. The similarity is a number between 0 and 1, evaluating the visual similarity between the term and its translation.

In order to better capture the letter transfigurations between the various languages, the similarity between two letters is not limited to 0 and 1, as in the traditional LCSS measure. We consider common letter transfigurations between languages; for highly dominant letter transitions we assign a similarity of 0.5. For example the letter ‘j’ in German commonly maps to ‘y’ in English. As in ‘ja’ $\rightarrow$ ‘yes’, or ‘jahr’ $\rightarrow$ ‘year’. Other dominant mappings that we consider are: ‘k’ $\rightarrow$ ‘c’ (e.g., architekt $\rightarrow$ architect) and ‘z’ $\rightarrow$ ‘c’ (e.g. sozial $\rightarrow$ social)

We illustrate this in the following example, where we compute the distance between the German word ‘demokratie’ and its English translation ‘democracy’. Before any comparison is done, diacretic marks (umlauts or accents) are removed and words are converted to lowercase. In Figure 5.3 one can notice that the similarity is increased by 0.5 when the letter k is compared to the letter c. Finally, the normalized similarity between the two words is $6.5/10 = 0.65$.

Naturally, due to polysemy issues (multiple meanings of a word) we need to evaluate the similarity with all possible translations and retain the best score. Let $\mathcal{T}(t, L)$ contain all translations of t in language L. Then, we define the cognativity of the term t as:

$$cogn(t, L) = \max_{\texttt{tr}(t) \in \mathcal{T}(t,L)} sim(t, \texttt{tr}(t, L))$$

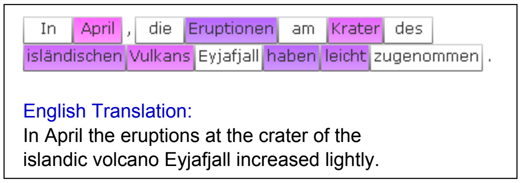


LCSS = demokra

English \ German	d	e	m	o	k	r	a	t	i	e
y	1	2	3	4	4.5	5.5	6.5	6.5	6.5	6.5
c	1	2	3	4	4.5	5.5	6.5	6.5	6.5	6.5
a	1	2	3	4	4.5	5.5	6.5	6.5	6.5	6.5
r	1	2	3	4	4.5	5.5	5.5	5.5	5.5	5.5
c	1	2	3	4	4.5	4.5	4.5	4.5	4.5	4.5
o	1	2	3	4	4	4	4	4	4	4
m	1	2	3	3	3	3	3	3	3	3
e	1	2	2	2	2	2	2	2	2	2
d	1	1	1	1	1	1	1	1	1	1

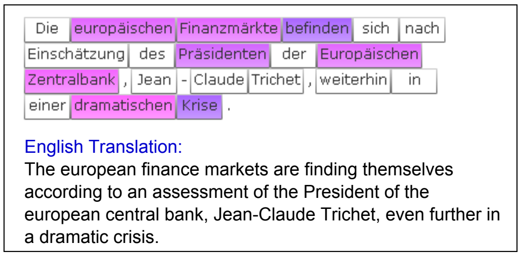
Figure 5.3: Evaluating the cognativity between demokratie (German) and democracy (English)

Examples: In the following examples we demonstrate the cognate identification ability of our algorithm. Words with lighter (more red) background color have higher cognativity score compared to words in darker background. Words not identified as cognates are displayed in white.



In April , die Eruptionen am Krater des isländischen Vulkans Eyjafjall haben leicht zugenommen .

English Translation:
In April the eruptions at the crater of the islandic volcano Eyjafjall increased lightly.



Die europäischen Finanzmärkte befinden sich nach Einschätzung des Präsidenten der Europäischen Zentralbank , Jean - Claude Trichet , weiterhin in einer dramatischen Krise .

English Translation:
The european finance markets are finding themselves according to an assessment of the President of the european central bank, Jean-Claude Trichet, even further in a dramatic crisis.

Figure 5.4 shows examples of German words with various cognativity scores with respect to English, as discovered by our method. The effectiveness of the cognate identification algorithm is evident. Naturally, this method can be used for other language pairs that originate from the same family, which is true for many of the European languages [102].

	German	English
	Spaghetti	spaghetti
	Graph	graph
	Astronaut	astronaut
	...	...
high cognativity	applikation	application
	Faszination	fascination
	Korruption	corruption
	Pyjama	pajama
	lithographisch	lithographic
	...	...
	Endokrinologin	endocrinologist
	Konservatismus	conservatism
	komfortabel	comfortably
	multipliziert	multiplied
low cognativity	Galerie	gallery
	...	...
	produktregel	product rule
	packmaterial	packing materials
	mobiltelefon	mobile phone
	...	...
	kompromisslos	uncompromising
	kalenderwoche	calendar week

Figure 5.4: Some identified German cognates

Combining Popularity and Cognativity

The identification of cognates is used in order to properly assess word familiarity, irrespective of its web popularity. In other words, cognativity is the dominant factor: if a term is the same (or almost the same) in the user's native language, then it is expected to be familiar even if the term is rarely used. We consider a word as a cognate if the cognativity value is greater than a cutoff threshold value ξ. For our experiments we set $\xi = 0.45$. We converged to this value using a cross-validation on the results of a relevant user-study. Cognates are assigned the maximum possible familiarity, equal to 1. The familiarity of non-cognates is assigned as equal to their popularity. Formally, we define the familiarity of a term t with respect to a language L as follows:

$$fam(t, L) = \begin{cases} pop(t), & \xi < cogn(t, L) \\ 1, & \xi \geq cogn(t, L) \end{cases} \tag{5.5}$$

Alternatively, one could assign a reduced (but still high) familiarity value for cognates that are less obvious. Illustratively, the German word 'faktor' (=factor) would map to a familiarity of 1, whereas the less apparent ‘brezel’ (=pretzel) could be assigned a familiarity of (for example) 0.75.

Equation 5.5 gives us the familiarity of a single term. We define the aggregate familiarity of an entire document by

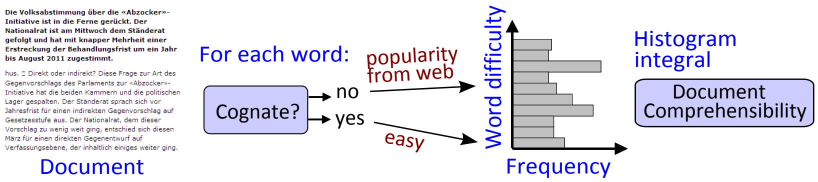
$$fam(d) \quad := \quad \sum_{t \in d} \frac{count(t,d)}{words(d)} fam(t,L), \tag{5.6}$$

where $count(t,d)$ is the total number of appearances of term t in document d, and $words(d) = \sum_{t \in d} count(t,d)$ is the total number of words in d. Instead of computing the familiarity of the document as a whole, one can do this separately for each of each parts (e.g. sentences or paragraphs). This might be preferable for very long or diverse documents, where the familiarity of the vocabulary may vary greatly.

An alternative approach is to first create a histogram of the familiarity scores of all the terms in the document; this will have the effect of acting as a smoothing operator. Therefore, the familiarity values are aggregated into N bins, $(fam_i, f_i), i = 1, \cdots, N$, where fam_i is the familiarity value assigned to the i-th bin, and f_i is the frequency (in the entire document) of the terms belonging to the i-th bin. Then:

$$fam(d) \quad := \quad \sum_{i=1}^{N} f_i \times fam_i. \tag{5.7}$$

This process is depicted graphically in Figure 5.5 which is the one also used in our experiments.



Die Volksabstimmung über die «Abzocker»-Initiative ist in die Ferne gerückt. Der Nationalrat ist am Mittwoch dem Ständerat gefolgt und hat mit knapper Mehrheit einer Erstreckung der Behandlungsfrist um ein Jahr bis August 2011 zugestimmt.

hus. ⌑ Direkt oder indirekt? Diese Frage zur Art des Gegenvorschlags des Parlaments zur «Abzocker»-Initiative hat die beiden Kammern und die politischen Lager gespalten. Der Ständerat sprach sich vor Jahresfrist für einen indirekten Gegenvorschlag auf Gesetzesstufe aus. Der Nationalrat, dem dieser Vorschlag zu wenig weit ging, entschied sich diesen März für einen direkten Gegenentwurf auf Verfassungsebene, der inhaltlich einiges weiter ging.

Figure 5.5: Computing the overall document comprehensibility from the individual word scores

5.4.3 Word Decompounding

One issue that we encountered when determining the proper familiarity value of a word is due to the presence of compound words. Several languages such as German, Dutch or Swedish, are known as *compounding languages* because they allow the creation of new complex words by merging together simpler ones. In that way, more complex concepts can be created through combination of nouns, verbs and adjectives. As an example, the compound word ‘Medizindoktor’ (=medical doctor) cannot be found in a dictionary and potentially also has few occurrences in texts or the web; however, its meaning is easily understandable given its building blocks.

It is therefore instructive to identify compound words and evaluate the individual familiarity of their components. The splitting of a compound word in its basic parts is called *decompounding*. Addressing word decompounding is an important issue for German texts [103], which are also the focus of our experiments. It has been noted that a large percentage of words in German texts are indeed compounds. For example, Schiller identified more than 40% of the words in a large German newspaper corpus as compounds [104].

Some examples of German compounds are provided below. We purposefully pick compound words that contain cognates for enhanced readability:

2-compounds:

Aschewolke (=ash clouds)

sozialdemokratie (=social democracy)

europaweit (= europe-wide)

3-compounds:

multiprozessorsystem (=multiprocessor system)

Jahrhundertwende (=hundred year turnpoint)

In our system we utilize a simple but effective algorithm for identifying 2- and 3-compounds. For ease of exposition, and with no serious loss in generality, we provide a treatment for detection of 2-compounds.

Given a term t of length n, let $sub(\alpha, i, j)$ return the substring of α that begins at position i (inclusive) and ends at position j (inclusive). If $i > j$, the function returns $\emptyset$. We define the *decompounded* familiarity of t with respect to a language L as follows:

$$\begin{aligned} fam^{DC}(t, L) = &\max_i \frac{1}{2}\{fam(sub(t, 1, i), u) \\ &+ fam(sub(t, i+1, n), u)\} \end{aligned}$$

when $1 \leq i \leq n$ and $fam(\emptyset) = 0$. Therefore, the above formula discovers the split point that maximizes the popularity of the two subcomponents. The final familiarity of the word is the maximum between the decompounded familiarity and original one (when treating the word as a whole).

As an example, in the texts of the previous section, the words Finanzmärkte and Zentralbank had higher familiarity when decompounded rather than when considered as non-compound words.

5.5 Tuning Comprehensibility Weights

The overall comprehensibility of a document is affected by the familiarity and readability of its terms. The effect of each of these two factors is controlled via the respective weights w_1, w_2. We discuss how these weights can be estimated from available user ratings.

Given is a set of k training documents $\mathcal{D}$ with known familiarity $fam(d, u)$, readability $rd(d_i)$ and a given rating $r_d(u) \in [0, 1]$ from user u for each $d \in \mathcal{D}_{tr}$, indicating the comprehensibility of document $d \in \mathcal{D}_{tr}$. In practice, these values can be simply obtained by asking users to rank the documents by comprehensibility, or to assign a discrete value taken from a predefined set of difficulty levels. These ratings can then be scaled to take values in $[0, 1]$, without loss in generality.

The goal is to minimize a cost function of the error between the comprehensibility function outcome for the given familiarity-readability values, and the user ratings. We may consider minimizing the *Minimum Square Error* (MSE), or the *Minimum Absolute Error* (MAE), subject to non-negativity constraints on the parameters w_1, w_2; the first problem is a quadratic program that has an explicit solution [105], while the second one can be easily mapped to a Linear Program (LP) and, consequently, solved numerically very efficiently using Interior-Point methods [105]. Here, we focus on another objective function, namely *Minimum Rating Order Violation* (MROV), where the goal is not to fit the the parameters to better predict user ratings, but rather seek to preserve the relative ordering of the given rankings as accurately as possible. Let us define $w := (w_1, w_2)^T$, where $w_1, w_2 \geq 0$ are non-negative weights, and $N := |\mathcal{D}_{tr}|$.

5.5.1 Minimum Rating Order Violation (MROV)

We consider the case where we are not interested in the absolute rating values, but only seek to estimate $w_1, w_2 \geq 0$ so that the relative ordering of the user ratings is preserved as well as possible. First, we sort the user ratings in decreasing order, whence $r(1) \geq r(2) \geq \cdots \geq r(N)$ and let the corresponding induced ordering of the documents be $d_1, \cdots, d_N$. In such case, we are looking for $w_1, w_2 \geq 0$ so that:

$$w_1 fam(d_1, u) + w_2 rd(d_1) \geq \cdots \geq w_1 fam(d_N, u) + w_2 rd(d_N)$$

Of course this might not be feasible except for the trivial case that $w_1 = w_2 = 0$. Additionally, the set of $w \geq 0$ that satisfies this set of inequalities is a *cone*, i.e., if $w \geq 0$ is in the feasible set, then λw is also in the feasible set for any $\lambda \geq 0$. This fact that the ordering constraints are *positively homogeneous* in w implies, in turn, that only the ratio $\frac{w_2}{w_1}$ is of actual importance. Therefore, we consider fixing one variable, say $w_1 = 1$, and optimizing over the other:

$$\begin{aligned} \min_{w} \quad & \textstyle\sum_{i=1}^{N-1} v_i && (5.8)\\ \text{s.t.} \quad & fam(d_i, u) + w \times rd(d_i) \geq && \\ & fam(d_{i+1}, u) + w \times rd(d_{i+1}) - v_i, && (5.9)\\ & i = 1, \cdots N-1 && \\ & v_i \geq 0 && (5.10)\\ & w \geq 0 && (5.11) \end{aligned}$$

For each $i = 1, \cdots N-1$, v_i denotes the amount of violation of the $i-$th inequality (5.9) [2]. The optimal solution w_{opt} of the LP can subsequently be rescaled to yield a desirable dynamic range for the rating function.

Remark 1 (Multiple rating users)**.** *In the case there are U users available for rating the documents, we can apply the above approach to the average rate $r(i) := \frac{1}{U}\sum_{u=1}^{U} r_i^{(u)}$, where $r_i^{(u)}$ denotes the rating of the $u-$th user for the $i-$th document. We can further consider the simple extension*

$$\begin{aligned} \min_{w} \quad & \textstyle\sum_{u=1}^{U} c_u (\sum_{i=1}^{N-1} v_{ui}) && (5.12)\\ \textit{s.t.} \quad & fam(d_{u_i}, u) + w \times rd(d_{u_i}) \geq && \\ & fam(d_{u_{i+1}}, u) + w \times rd(d_{u_{i+1}}) - v_{ui}, && (5.13)\\ & u = 1, \cdots, U, \quad i = 1, \cdots N-1 && \\ & v_{ui} \geq 0 && (5.14)\\ & w \geq 0, && (5.15) \end{aligned}$$

[2] We consider non-negative variables v_i, since we are interested in the case where not all ordering-imposed inequalities can be simultaneously fulfilled. If there exist weights such that all inequalities can be satisfied, letting v_i's unconstrained also yields the amount of required *slackness*.

where c_u is a weight used to denote the confidence in user u's ratings, and where for each user u documents are sorting in descending order of rating with corresponding indices $u_i, i = 1, \cdots, N$.

Finally, we can also consider fixing $w_2 = 1$ and optimize with respect to $w = w_1 \geq 0$ as before, and compare the MROV values obtained from the two approaches.

5.6 Skyline Ranking

We have shown a way to evaluate the comprehensibility of a document for non-native users. The relevance of the document can also be evaluated using standard techniques like those presented in Section 5.2. Therefore, each document is represented by a two-dimensional vector $(fam(d), rel(d)) \in \mathbb{R}_+^2$, where $fam(d), rel(d)$ denote the document's familiarity and relevance, respectively. Naturally, there are documents which present a higher rating in comprehensibility, relevance or both attributes and *dominate* other documents. Those represent the *skyline* which we denote by the set $\mathcal{S} \subset \mathcal{D}$ (or Pareto boundary) for a set of documents $\mathcal{D}$. There exist efficient ways of calculating the skyline [55, 106].

Once the skyline is computed, documents can be presented in different order to the user. For example, one can start by presenting first the document lying on the *middle* of the skyline, which represents a document of average difficulty and relevance. How one defines the middle document depends on the final utility function F_u, which for example can be the area of dominance under the document/point. In this case, $F_u(fam, rel) = fam \times rel$, or $F_u(fam, rel) = v_1 \times fam + v_2 \times rel, v_1, v_2 \geq 0$. This is illustrated in Figure 5.6.

In this manner, the user can navigate on the skyline by scrolling to either more relevant (left side) or more comprehensible (right side) documents. Last but not least, this interface can assist in the evaluation of the relative comprehensibility and relevance of *any* document on the search result, based on its distance from the skyline points.

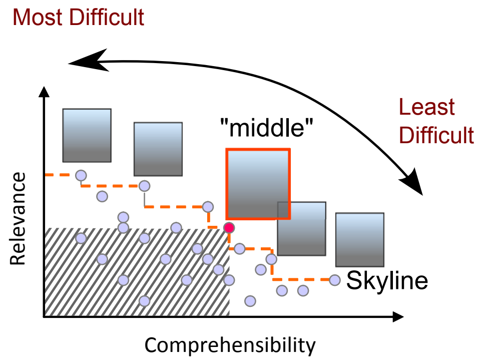


Figure 5.6: Navigating on the skyline of top-rated results

5.7 Experiments

In this section we illustrate the ability of our approach to capture the inherent comprehensibility of foreign textual content. We mainly focus on German texts. We start with a small user study and then proceed to a larger corpus.

5.7.1 User Study

Initially, we want to estimate how well the proposed comprehensibility measure approximates the ranking provided by human annotators. We avoid including texts of different topics, since this may introduce some bias. Therefore we have assembled documents that address the same topic but examine different aspects of it and possibly addressing different audiences. The topic we have focused on is the financial crisis in Greece during 2010. In order to include texts of variable comprehensibility, we have selected texts from sources with a consistent language level. For each language, we first selected 3 segments from financial web sites (e.g. bloomberg.com); these approach the topic from a more technical view point, thus employing a more sophisticated and formal language. We then selected three segments from articles from popular news-portals (e.g. reuters.com); since these are addressed to the general public, the language used is well-structured, with an average level of sophistication. Finally, three texts were taken from relevant comments posted in public forums by users; the language in these segments is generally simpler and informal, lacking sophisticated terms and constructs.

The nine German texts were given to eight human annotators who are not native speakers but possess a reasonable command of the German language. In addition, all the annotators were chosen to be native or proficient in the English language, in order to enhance the effect of cognativity. Similarly, we have repeated an equivalent test using nine English texts, which were given to eight annotators native in German, with good command of the English language.

German Texts		
ours	user Avg	Std
1	1	0
2	2.8	0.9
3	3	0.82
4	3.4	1.3
5	5.9	1.25
6	6.7	1.28
7	6.9	0.99
8	6.3	1.28
9	9	0

English Texts		
ours	user Avg	Std
1	1.5	0.53
2	2.38	1.41
3	3.25	1.28
4	4.63	1.41
5	4.88	1.55
6	7.38	1.85
7	6.88	1.55
8	6.13	2.47
9	8	0.76

Figure 5.7: User Study on German and English texts: texts are ranked by our technique, as well as by human annotators.

The annotators were asked to rank the texts from easiest to most difficult. We also computed the scores for each text, using the developed comprehensibility formula. The results are shown in Figure 5.7.1. The first column of each table shows the rank of each text based on the scores assigned by our method, the second and third columns hold the average rating and the standard deviation assigned from the annotators, respectively.

The results of the study are very promising. For both the German and English texts, the rank based on our method and the average human rating were consistently very close. Our methodology was successful in ranking the texts by comprehensibility, illustrating its potential usefulness in the context of foreign document retrieval. For the experiment with German documents, the observed standard deviation values were consistently low, indicating a strong consensus among the annotators. The respective values for English were slightly elevated, indicating that this task was more challenging and the annotators were not always in agreement. Nonetheless, our comprehensibility formula was still able to capture the average rating assigned by the annotators. In order to provide some qualitative evidence, we present a segment of the texts placed first, last and in the middle of the rankings given by our method. We provide excerpts from the English text for ease of exposition:

- **Ranked #1:** "You could only avoid the big cities during riots. The financial crisis does not affect travelers. It only affects Greek citizens for now."
- **Ranked #5:** "Some say the key to stabilizing the Greek, Portuguese and Spanish economies is for those debt-plagued countries to remove themselves from the common currency."
- **Ranked #9:** "That said, there are also substantial risks to the program, including the possibility of weaker-than-estimated economic growth, political and public fatigue for implementing the steps, and deficiencies in fiscal data, the fund said."

Readability vs Familiarity: Next, we demonstrate how the two components of our comprehensibility measure perform on their own. This is shown in Figure 5.8 where we juxtapose familiarity with the Flesch readability measure. On the x-axis we plot the rank of each document according to either familiarity or readability and then on the y-axis is the average score given by users. We observe that the proposed familiarity measure is better at capturing the users' notion and demonstrates smaller variance. Readability fails to capture the difficulty of the documents. This is easy to comprehend, since readability measures mostly capture the difficulty due to sentence structure, but not due to vocabulary, which is more important for foreign documents. Structure is secondary, and becomes important only once someone has become more acquainted with the language.

5.7.2 Large-Scale Evaluation on Real Data

While our study provided valuable insight to the efficacy of our approach, a large-scale evaluation is still required to solidify our findings. We use the data provided by the edu-

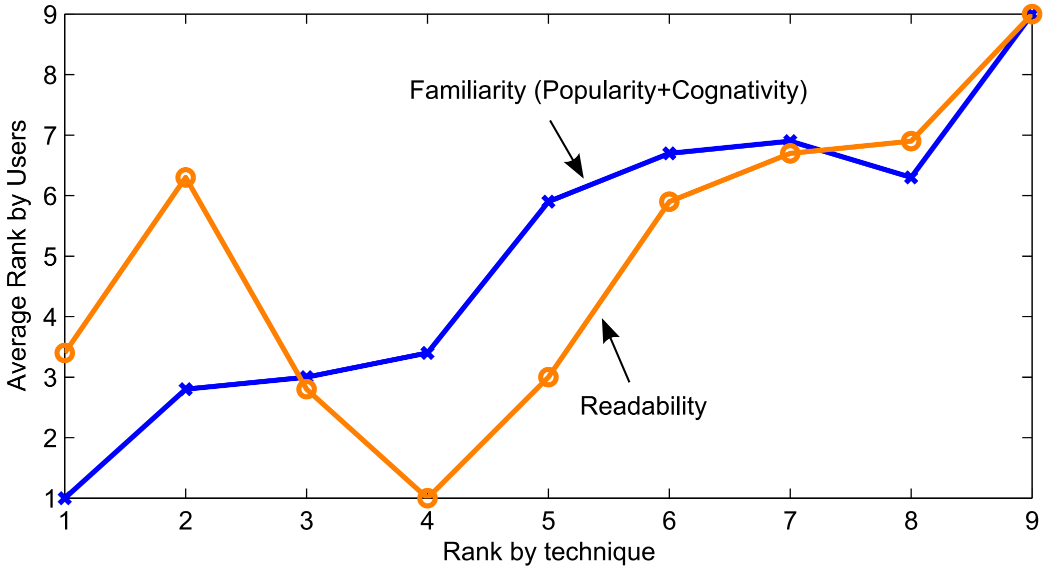


Figure 5.8: Comparing Familiarity and Readability measures. Readability fails to capture the document's true difficulty.

cational website `CourseInfo.com`, which hosts essays on a variety of topics, including foreign languages. The purpose of the site is to provide reading material of variable difficulty levels for students native in English. In particular, essays are grouped into 3 levels of increasing difficulty: GCSE (300 essays for high school students), A-level (150 essays for pre-college preparation) and University-level (50 essays for Bachelor-level students). In our experiment, we use all the available essays from the "German Essays" category.

For the first part of our evaluation, we use our approach to measure the comprehensibility of each essay, using an equal weight for readability and familiarity. As mentioned above, each essay belongs to one of three difficulty levels: `A-Level`, `GCSE` or `University`. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be the sets of essays corresponding to two of the three levels and assume that $\mathcal{D}_1$ corresponds to a level easier than D_2 (e.g. $\mathcal{D}_1$ has the essays from `GCSE` and $\mathcal{D}_2$ from `University`). Then, the observed error percentage for this pair is:

$$error(\mathcal{D}_1, \mathcal{D}_2) = \frac{|\{d_1, d_2) : d_1 \in \mathcal{D}_2, d_2 \in \mathcal{D}_2, C(d_1) < C(d_2)\}|}{|\mathcal{D}_1| \times |\mathcal{D}_2|} \tag{5.16}$$

The error is defined as the fraction of possible essay-pairs (d_1, d_2), where $d_1 \in \mathcal{D}_1$ and

$d_2 \in \mathcal{D}_2$ and d_1 has received a lower comprehensibility score by our approach than d_2. This is undesirable, since Eq. 5.16 assumes that $\mathcal{D}_1$ corresponds to an easier level than $\mathcal{D}_2$. The computed error values for all possible level-combinations are shown in Table 5.7.2.

Table 5.1: Observed error for CourseInfo Data

Confusion Matrix		
Levels	A-level	University
GCSE	13.7%	3.1%
A-level		27.5%

Observe that for `GCSE` and `University` (the two levels that differ the most in terms of difficulty) the observed error was minimal (3.1%). A small error was also observed for the `GCSE` and `A-Level` pair, indicating that our approach can consistently distinguish `GCSE` essays. The highest error was observed for the `A-Level`/`University` pair. An inspection of some of the erroneous pairs revealed that deducing the true level of difficulty was an ambiguous task, even for a human annotator. Still, as shown in the table, such pairs made up for less than a third of the total. In short, our approach performed consistently well, managing to detect the, often subtle, gap in comprehensibility among the three levels.

Readability vs Familiarity: Finally, we demonstrate that familiarity is is a more robust estimator of a document's comprehensibility than readability. Figure 5.9 plots the readability and familiarity of the 3 classes of documents from CourseInfo, in descending order. Even though both measures provide accurate class distinction, familiarity is clearly a more *robust* estimator, since it introduces very little in-class variance. In particular, For the the `GCSE`, `A` and `University` levels, the variance of the readability score was 0.002, 0.005 and 0.005, respectively. The corresponding values for familiarity 0.0005, 0.0008 and 0.0006. In general, as described in the previous sections, according to the application at hand, it is instructive to merge the two measures. Both measures offer a different view of a document's difficulty. However, in foreign documents, more weight should be given in the vocabulary aspect of a document, which is crystallized in the proposed familiarity measure.

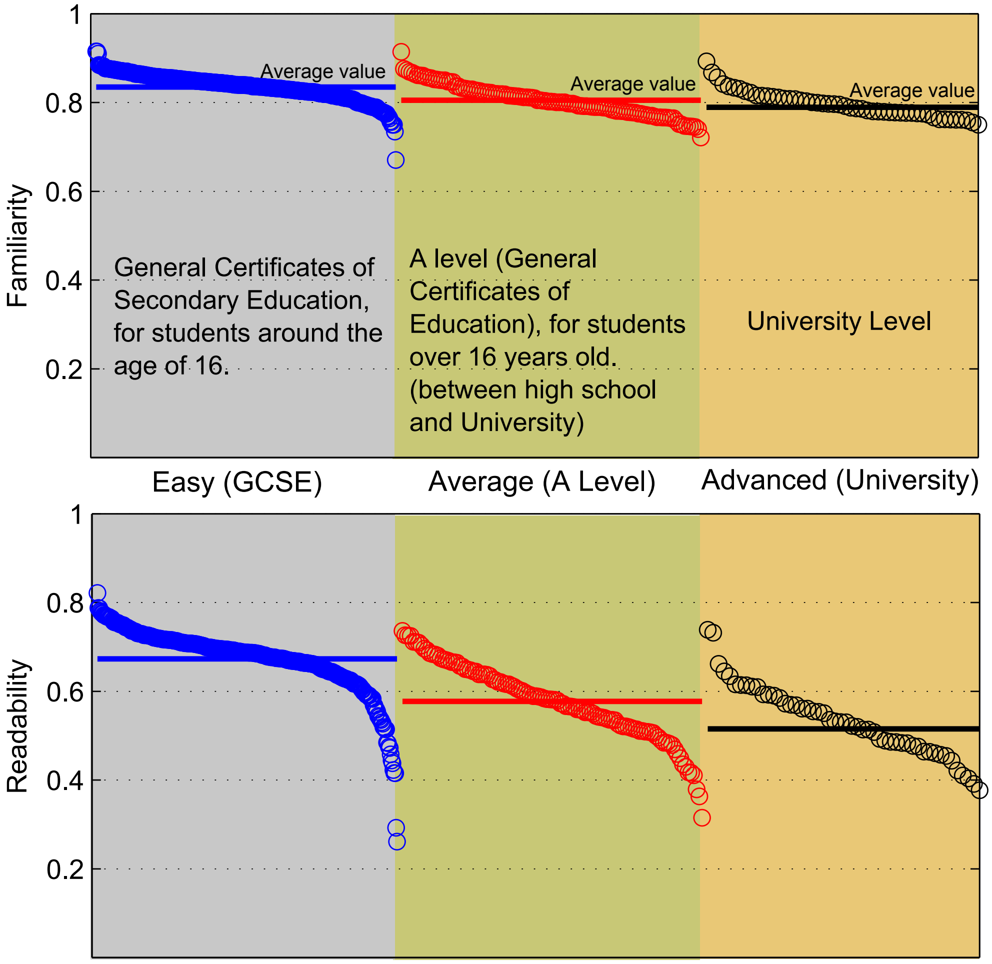
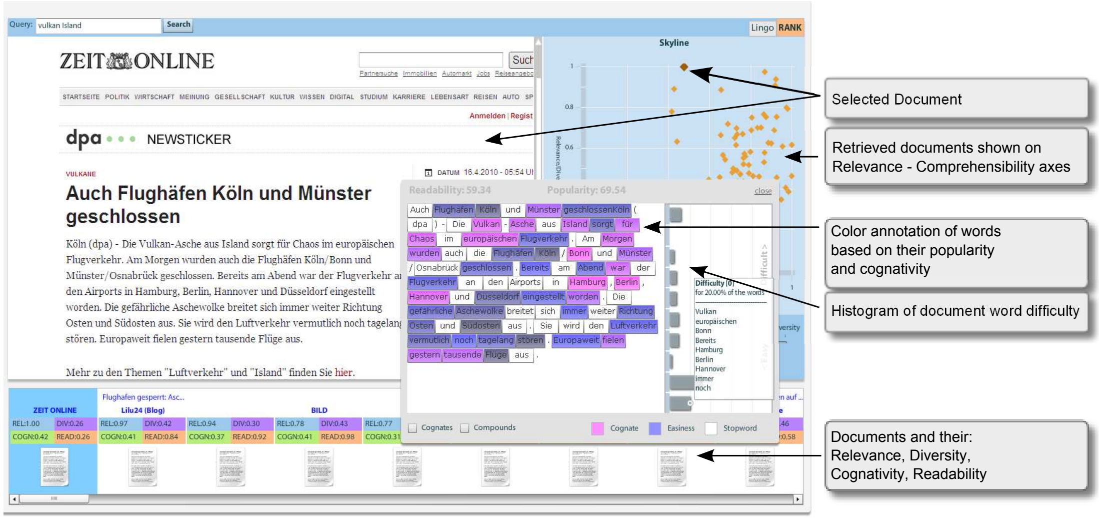


Figure 5.9: Comparing Familiarity and Readability. Familiarity is a more robust estimator of the document's difficulty.

At a glance, it may that the computed familiarity values among the three levels are not always great. This is due to the fact that, even in some harder documents, the majority of the words can often be familiar (e.g. connector-words, frequent verbs etc). However, it is the rest of the words that convey the document's meaning and define its overall familiarity. This is captured by our approach which, as clearly demonstrated by the previous experiment, consistently assigns higher scores to documents from easier levels.

5.7.3 LingoRank GUI

Around our technology we have built a web application, called `LingoRank`[3], for the retrieval of German news stories targeting native English speakers. The application utilizes the Google News web API for retrieving the first superset of news documents relevant to a query. Subsequently, our application re-ranks them based on their estimated comprehensibility for English speakers.


Query: vulkan Island

ZEIT ONLINE

dpa NEWSTICKER

VULKANE

DATUM 16.4.2010 - 05:54 Uh

Auch Flughäfen Köln und Münster geschlossen

Köln (dpa) - Die Vulkan-Asche aus Island sorgt für Chaos im europäischen Flugverkehr. Am Morgen wurden auch die Flughäfen Köln/Bonn und Münster/Osnabrück geschlossen. Bereits am Abend war der Flugverkehr an den Airports in Hamburg, Berlin, Hannover und Düsseldorf eingestellt worden. Die gefährliche Aschewolke breitet sich immer weiter Richtung Osten und Südosten aus. Sie wird den Luftverkehr vermutlich noch tagelang stören. Europaweit fielen gestern tausende Flüge aus.

Mehr zu den Themen "Luftverkehr" und "Island" finden Sie hier.

Readability: 59.34 Popularity: 69.54

Auch Flughäfen Köln und Münster geschlossenKöln (dpa) - Die Vulkan - Asche aus Island sorgt für Chaos im europäischen Flugverkehr . Am Morgen wurden auch die Flughäfen Köln / Bonn und Münster / Osnabrück geschlossen . Bereits am Abend war der Flugverkehr an den Airports in Hamburg , Berlin , Hannover und Düsseldorf eingestellt worden . Die gefährliche Aschewolke breitet sich immer weiter Richtung Osten und Südosten aus . Sie wird den Luftverkehr vermutlich noch tagelang stören . Europaweit fielen gestern tausende Flüge aus .

Difficulty [0]
for 20.00% of the words
Vulkan
europäischen
Bonn
Bereits
Hamburg
Berlin
Hannover
immer
noch

ZEIT ONLINE		Lilu24 (Blog)				BILD		
REL:1.00	DIV:0.26	REL:0.97	DIV:0.42	REL:0.94	DIV:0.30	REL:0.78	DIV:0.43	REL:0.77
COGN:0.42	READ:0.26	COGN:0.41	READ:0.84	COGN:0.37	READ:0.92	COGN:0.41	READ:0.98	COGN:0.31

Search
Lingo RANK
Suche
Partnersuche Immobilien Automarkt Jobs Reiseangebo
STARTSEITE POLITIK WIRTSCHAFT MEINUNG GESELLSCHAFT KULTUR WISSEN DIGITAL STUDIUM KARRIERE LEBENSART REISEN AUTO SP
Anmelden | Regist
Skyline
1
0.8
0.6
Relevance/Dive
Difficult >
< Easy
close
versity
Flughafen gesperrt: Asc...
n auf ...
.46
0.58
□ Cognates □ Compounds
Cognate
Easiness
Stopword
Selected Document
Retrieved documents shown on Relevance - Comprehensibility axes
Color annotation of words based on their popularity and cognativity
Histogram of document word difficulty
Documents and their: Relevance, Diversity, Cognativity, Readability

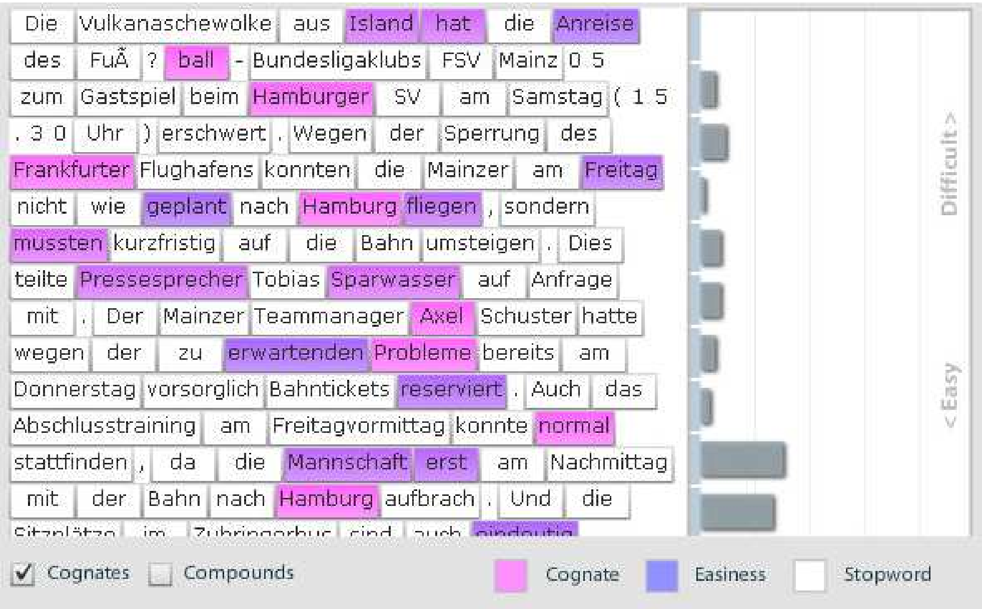
Figure 5.10: LingoRank: A tool for the retrieval and ranking of foreign news documents.

Figure 5.10 illustrates the interface, consisting of 3 panes: the bottom part aggregates the retrieved news documents; the top left depicts the selected document, and the top right plots the documents in 2 dimensions: relevance and comprehensibility of the text. Relevance to a query is estimated using a cosine similarity metric, combined with a basic measure of *diversity* based on the similarity of each document with the others. Comprehensibility is computed as described in the previous sections. Additionally, the interface provides the functionality of highlighting the identified cognates and word compounds using color annotations.

The user is first shown the documents on the skyline of the relevance-comprehensibility axes, but can also navigate to any other documents. The document selected in Figure 5.10 is

[3] `http://www.youtube.com/watch?v=jHiZQ9OOLg4`

one of the many returned on the German query **“Island Vulkan”** (Island Volcano) referring to the explosion of the Volcano in Island during the 1st quarter of 2010.



Die Vulkanaschewolke aus Island hat die Anreise
des FuÃ ? ball - Bundesligaklubs FSV Mainz 0 5
zum Gastspiel beim Hamburger SV am Samstag (1 5
. 3 0 Uhr) erschwert . Wegen der Sperrung des
Frankfurter Flughafens konnten die Mainzer am Freitag
nicht wie geplant nach Hamburg fliegen , sondern
mussten kurzfristig auf die Bahn umsteigen . Dies
teilte Pressesprecher Tobias Sparwasser auf Anfrage
mit . Der Mainzer Teammanager Axel Schuster hatte
wegen der zu erwartenden Probleme bereits am
Donnerstag vorsorglich Bahntickets reserviert . Auch das
Abschlusstraining am Freitagvormittag konnte normal
stattfinden , da die Mannschaft erst am Nachmittag
mit der Bahn nach Hamburg aufbrach . Und die
Sitzplätze im Zubringerbus sind auch eindeutig

☑ Cognates □ Compounds 🟪 Cognate 🟦 Easiness □ Stopword

Figure 5.11: Highlighting the identified *cognates* in `LingoRank`

Figure 5.11 shows the ability of LingoRank to indicate the comprehensibility and cognativity of individual words. In addition, the right side of the panel summarizes the comprehensibility of the whole document via a histogram view. In the Figure, we have selected to highlight the identified *cognates*. A stronger shade of red indicates higher cognativity with respect to English. For example, words like ‘Probleme’ (problems) or ‘normal’, demonstrate high cognativity scores, while others like ‘Fussball’ (football), are still cognates and easily understandable but are depicted in a darker shade of red because they carry lower cognativity scores.

5.8 Conclusion and Future Work

In this work we introduced a *sorting* operator that ranks foreign documents according to their perceived comprehensibility targeted to non-native speakers. We have successfully applied our methodology to the particularly challenging case of German language documents.

The proposed method can be effectively applied in applications such as: a) e-bookstores and education, for providing personalized book/text recommendation according to the appropriate language comprehensibility level, b) search engine refinement, for proper ranking of multilingual results. In addition, such technology can also be used for deciding *when* to perform document translation [107], i.e., *only* when a retrieved document is deemed extremely difficult to comprehend. Preliminary experiments are very encouraging, demostrating the applicability of our approach. In the immediate future, we plan to provide an extended evaluation of our techniques for other European languages.

Chapter 6

Finding a Team of Experts in Social Networks

6.1 Introduction

The success of a project depends not only on the expertise of the people who are involved, but also on how effectively they collaborate, communicate and work together as a team. Assume, for example, an IT project manager who wants to build a team of engineers skilled in the following areas: T=$\{$*algorithms, software engineering, distributed systems, web programming*$\}$. Also assume there are five candidates, $\{$*a, b, c, d, e*$\}$, with the following backgrounds: X_a=$\{$*algorithms*$\}$, X_b=$\{$*web programming*$\}$, X_c=$\{$*software engineering, distributed systems*$\}$, X_d=$\{$*software engineering*$\}$ and X_e=$\{$*software engineering, distributed systems, web programming*$\}$. The relationships among these candidates are represented by the social network shown in Figure 6.1, where the existence of an edge between two nodes in G indicates that the corresponding persons can collaborate effectively.

Without considering how effectively these people can collaborate, the manager can select either $\mathcal{X}' = \{a, b, c\}$ or $\mathcal{X}'' = \{a, e\}$, since both these teams have the required skillset. However, the existence of graph G makes $\mathcal{X}' = \{a, b, c\}$ a superior solution, since

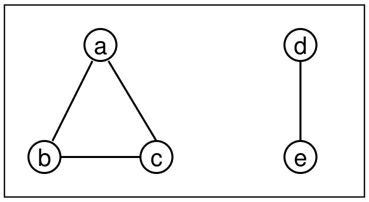


Figure 6.1: Network of connections between individuals in $\{a, b, c, d, e\}$.

the structure of G indicates that a and e cannot work together at all.

The existence of a social network between individuals is quite common in real scenarios. In a company, the network may capture the hierarchical organization of the employees. In this case, the graph encodes the fact that people in the same group or department can communicate easier than people working in different divisions. In a research community, the network captures previous successful collaborations among scientists. Other examples of social networks between professionals include LinkedIn (`www.linkedin.com`), Xing (`www.xing.com`) and others.

The problem: In this work, we study the problem of finding a group of individuals who can function as a team to accomplish a specific task. We assume that there exists a pool of n candidates $\mathcal{X} = \{1, \ldots, n\}$, where each candidate i has a set of skills X_i. We also assume that these candidates are organized in a *weighted* and *undirected* social graph $G\left(\mathcal{X}, E\right)$. The weights on the edges of G should be interpreted as follows: a low-weight edge between nodes i, j implies that candidate i and j can collaborate and/or communicate more easily than candidates connected with a high-weight edge. These weights can be instantiated in different ways in different application domains. For example, in a company, the weight between two employees may correlate to the length of the path from one employee to another through the organizational chart. In a scientific research community, the weight between two scientists is related to the total number of publications they have coauthored. Interpersonal relationships among individuals can also be used to calculate the weights.

Given a task T that requires a set of skills, our goal is to find a set of individuals $\mathcal{X}' \subseteq \mathcal{X}$, such that every required skill in T is exhibited by at least one individual in $\mathcal{X}'$. Additionally, the members of team $\mathcal{X}'$ should define a subgraph in G with *low communication cost*. The communication cost measures how effectively the team members can collaborate: the lower the communication cost, the better the quality of the team.

Our contributions: To the best of our knowledge, we are the first to consider the TEAM FORMATION problem in the presence of a social network of individuals. We study two instances of this problem, analyze them rigorously and present algorithms for their solution. Our experiments illustrate that our problem definitions, as well as our algorithms, work well in practice and give useful and intuitive results.

Roadmap: The rest of this work is organized as follows: in Section 6.2 we review the related work on team formation and task allocation. In Section 6.4 we formally define the TEAM FORMATION problem and identify the two variants that we are going to consider in this work. In the same section, we also study their computational complexity. In Section 6.5 we give algorithms for the different variants of the TEAM FORMATION problem and in Section 6.6 we illustrate the usefulness of our methodology on a real collaboration dataset. We conclude in Section 6.7.

6.2 Related work

There is a considerable amount of literature on TEAM FORMATION in the operations research (OR) community [108, 109, 110, 111]. A trend in this line of work is to formulate the TEAM FORMATION problem as an integer linear program (ILP), and then focus on finding an optimal match between people and the demanded functional requirements. The problem is often solved using techniques such as simulated annealing [108], branch-and-cut [110] or genetic algorithms [111]. The main difference between the studies above and our work is that we explicitly take into account the social graph structure of the individuals, in the process of team formation. In most of the previous work, the organizational or social bonds among individuals are ignored and the focus is limited on their skills. Moreover, the problem formulations we provide, and the algorithmic approaches we take, are fundamentally different from those proposed in the OR literature.

The necessity of effective collaboration among individuals in a team has been considered in the past. Fitzpatrick and Askin [112] use the Kolbe Conative Index (KCI) to measure individuals' drive and temperament, which in turn reflects the quality of the team. Chen and Lin [109] use the Myers-Briggs test to measure the candidates' personality and evaluate their interpersonal relationships as team members. Although these approaches are interesting from the anthropological/psychological point of view, they also ignore the existing graph structure among individuals. Therefore, these approaches should be considered complementary to ours.

The network structure between individuals in a workforce pool has been taken into account by Gaston *et al.* [113]. The authors provide an experimental study of how different graph structures among the individuals affect the performance of a team. Although related, the work presented in [113] does not address the computational problem of finding a team of experts in a given network. Some work has also been devoted to the construction of the social network [114, 111], given a pool of skilled individuals.

The dynamics of group-formation processes and their impact on the formation of communities in networks have been recently addressed in [115]. The game-theoretic aspects of the same problem have been studied in [116]. These studies are complementary to ours and mostly focus on providing useful insights about social processes.

6.3 Preliminaries

We assume a pool of candidates consisting of n individuals, $\mathcal{X} = \{1, \ldots, n\}$. We also assume $\mathcal{A} = \{a_1, \ldots, a_m\}$ to be a universe of m skills. Each individual i is associated with a set of skills $X_i \subseteq \mathcal{A}$. If $\alpha_j \in X_i$ we say that individual i *has* skill a_j; otherwise individual i does not have skill a_j. We often use the set of skills an individual possesses to refer to him. Also, we say that a subset of individuals $\mathcal{X}' \subseteq \mathcal{X}$ possesses skill a_j if there exists at least one individual in $\mathcal{X}'$ that has a_j.

A *task* T is simply a subset of skills required to perform a job. That is, $T \subseteq \mathcal{A}$. If $a_j \in T$ we say that skill a_j is *required* by task T. We can also define the *cover* of a set of individuals $\mathcal{X}'$ with respect to task T, denoted by $C\left(\mathcal{X}', T\right)$, to be the set of skills that are required by T and for which there exists at least one individual in $\mathcal{X}'$ that has them. That is, $C\left(\mathcal{X}', T\right) = T \cap \left(\cup_{i \in \mathcal{X}'} X_i \right)$. Given a skill $a \in \mathcal{A}$, we define its *support set* (or simply *support*), denoted by $S\left(a\right)$, to be the set of individuals in $\mathcal{X}$ that has this skill. That is, $S\left(a\right) = \{i \mid i \in \mathcal{X} \text{ and } a \in X_i\}$.

As we have already discussed, we assume that individuals are organized in an *undirected* and *weighted* graph $G\left(\mathcal{X}, E\right)$. Every node of G corresponds to an individual in $\mathcal{X}$; E is the set of edges connecting the nodes. The edges of G are weighted; edges of low (high) weight represent low (high) communication cost between the nodes they connect. Without loss of generality, we assume that the graph G is connected; we can transform every disconnected subgraph to a connected one, by simply adding edges with very high weight between every pair of nodes that belong to different connected components. Note that this very high weight is a number higher than the sum of all pairwise shortest paths in G.

For every two nodes $i, i' \in \mathcal{X}$ we define the *(graph) distance* function $d(i, i')$ to be the weight of the shortest path between i and i' in G. Note that this distance function between the nodes is a metric and thus satisfies the triangle inequality. For every pair of nodes we also use *Path*(i, i') to represent the set of nodes that are along the shortest path from i to i'. Apart from computing the distance between two nodes in G, we will often need the distance between a node $i \in \mathcal{X}$ and a set of nodes $\mathcal{X}' \subseteq \mathcal{X}$. We define this to be $d\left(i, \mathcal{X}'\right) = \min_{i' \in \mathcal{X}'} d\left(i, i'\right)$. In this case, we use *Path*$(i, \mathcal{X}')$ to represent the set of nodes that are along the shortest path from i to the node $j = \operatorname{argmin}_{i' \in \mathcal{X}'} d(i, i')$.

Finally, given graph G and $\mathcal{X}' \subseteq \mathcal{X}$, we use $G\left[\mathcal{X}'\right]$ to denote the subgraph of G that contains only the nodes in $\mathcal{X}'$.

6.4 Problems

In this section, we formally define the TEAM FORMATION problem that we address in this work. Our problem definitions reflect our belief that efficient communication among team members is an important factor for the successful completion of a task.

6.4.1 Problem Definition

Problem 8. *[*TEAM FORMATION*] Given the set of n individuals $\mathcal{X} = \{1, \ldots, n\}$, a graph $G(\mathcal{X}, E)$, and task T, find $\mathcal{X}' \subseteq \mathcal{X}$, so that $C(\mathcal{X}', T) = T$, and the* communication cost CC $(\mathcal{X}')$ *is minimized.*

In order to stress the generality of the TEAM FORMATION problem, we have deliberately avoided defining the communication cost in the definition of Problem 8. In this work, we focus on two instantiations of the communication-cost function. We chose these instantiations as we believe they are practical, simple and intuitive.

Diameter (R)**:** Given graph $G(\mathcal{X}, E)$ and a set of individuals $\mathcal{X}' \subseteq \mathcal{X}$, we define the *diameter communication cost* of $\mathcal{X}'$, denoted by CC-R $(\mathcal{X}')$, to be the diameter of the subgraph $G[\mathcal{X}']$. Recall that the diameter of a graph is the largest shortest path between any two nodes in the graph.

Minimum Spanning Tree (MST)**:** Given graph $G(\mathcal{X}, E)$ and $\mathcal{X}' \subseteq \mathcal{X}$ we define the MST *communication cost* of $\mathcal{X}'$, denoted by CC-MST $(\mathcal{X}')$, to be the cost of the *minimum spanning tree* on the subgraph $G[\mathcal{X}']$. Recall that the cost of a spanning tree is simply the sum of the weights of its edges.

We call the TEAM FORMATION problem with communication function CC-R, the

DIAMETER-TF problem. Similarly, we refer to the TEAM FORMATION problem with communication function CC-MST as the MST-TF problem.

Proposition 2. *The* DIAMETER-TF *problem is NP-complete.*

Proof. We prove the proposition by a reduction from the MULTIPLE-CHOICE COVER (MCC) problem [117]. An instance of the MCC problem consists of a universe $V = \{1, \ldots, N\}$ of N elements, a $N \times N$ symmetric real matrix D with non-negative entries, and a $\mathcal{S} = \{S_1, \ldots, S_k\}$ such that each $S_i \subseteq V$. Given constant K, the decision version of the MCC problem asks whether there exists $V' \subseteq V$ such that for every $i \in \{1, \ldots, k\}$, $|V' \cap S_i| > 0$ and $\max_{(u,v) \in V' \times V'} D(u, v) \leq K$.

We transform an instance of the MCC problem to an instance of the DIAMETER-TF problem as follows: for every set S_i in the MCC problem we create a skill a_i. The task T to be performed requires all the k skills. That is, $T = \{a_1, \ldots, a_k\}$. For every element $v \in V$ of the MCC instance, we create an individual i_v with skills $X_v = \{a_i \mid v \in S_i\}$. Two individuals i_v and i'_v are connected in the graph G by an undirected edge with weight equal to $D\,(v, v')$. Given this mapping it is easy to show that there exists a solution to the MCC problem with cost at most K if and only if there exists a solution to the DIAMETER-TF problem with CC-R cost at most K. The problem is trivially in NP. □

Note that the above reduction does not assume anything about the distance function between the nodes in G. However, from [117], we know that the MCC problem is NP-hard even when the distance matrix D corresponds to a metric. Therefore, the DIAMETER-TF problem is NP-hard when the distance function d between the individuals in G is a metric. Observe that the above reduction is *approximation preserving*. Therefore, the approximation properties of the MCC problem described in [117] carry over to the DIAMETER-TF problem as well.

For the MST-TF problem, we have the following hardness result:

Proposition 3. *The* MST-TF *problem is NP-complete.*

Proof. We prove the proposition by a reduction from the GROUP STEINER TREE (GST) problem [118]. An instance of the GST problem consists of an undirected graph $G\,(V, E)$, cost function $c : E \to \mathbf{R}$ and k subsets of vertices (called groups) $\{g_1, \ldots, g_k\}$ with $g_i \subseteq V, i \in \{1, \ldots, k\}$.

Given constant K, the decision version of the GST problem asks whether there exists a subtree $T\,(V', E')$ of $G\,(V, E)$ (i.e., $V' \subseteq V$ and $E' \subseteq E$) such that $|V' \cap g_i| > 0$ for every $i \in \{1, \ldots, k\}$ and cost $\sum_{e \in E'} c(e) \leq K$.

We transform an instance of the GST problem to an instance of the MST-TF problem as follows: for every group g_i in the GST problem we create a skill a_i. The task T to be performed requires all the k skills. That is, $T = \{a_1, \ldots, a_k\}$. For every node $v \in V$ of the GST problem we create an individual i_v with skills $X_v = \{a_i \mid v \in g_i\}$. The graph G' of the MST-TF problem is identical to the graph G of the GST problem, where the cost function c determines the weights of the edges in the MST-TF instance of the problem. Given this mapping it is easy to show that there exists a tree solution to the GST problem with cost at most C if and only if there exists a solution to the MST-TF problem with CC-MST cost at most C. The problem is trivially in NP. □

As before, note that the proofs above do not assume anything about the distance function between individuals in G. However, since the GST problem remains NP-hard even when the graph edge weights satisfy the triangle inequality, so does the MST-TF. As in the case of the DIAMETER-TF problem, the above reduction is approximation preserving. Therefore, the approximation properties of the GST problem ([119] and references therein) carry over to the MST-TF problem as well.

6.4.2 Discussion

In the definition of the TEAM FORMATION problem and its specializations, we focused on minimizing the communication cost among team members. Other notions of the “effectiveness" of a team can lead to different optimization functions. For example, if the communication cost was not a concern, we could define as our goal to find $\mathcal{X}' \subseteq \mathcal{X}$, such that $C(\mathcal{X}', T) = T$ and $|\mathcal{X}'|$ is minimized. Such a problem definition ignores the existence of the underlying graph $G(\mathcal{X}, E)$, and is actually an instance of the classic SET COVER problem, which can be solved by the standard `GreedyCover` algorithm. Details are presented in Sections 6.5.2 and 6.6.

Optimizing both the cardinality of the team and the communication cost between its members would require the minimization of a function of the form $\alpha \cdot |\mathcal{X}'| + (1 - \alpha) \cdot \text{CC}(\mathcal{X}', G)$, where $\alpha \in [0, 1]$. For $\alpha = 1$ the problem seeks for teams with the minimum cardinality. For $\alpha = 0$ this problem is the TEAM FORMATION problem. However, for values of α in $(0, 1)$ it is not clear that optimizing this alternative function makes sense; this is mostly because the two terms in the sum are in different scales and there is no knowledge on how these scales relate.

Alternatively, these two objectives (team size and communication cost) could be taken into account simultaneously by defining the problem as a bi-objective optimization problem. In such cases the goal is to find *Pareto-optimal solutions* [120]. Note that a solution is called Pareto-optimal if there does not exist another solution that is better in both objectives. For many problems, the set of Pareto-optimal solutions is exponential to the size of the input and thus cannot be found in polynomial time. Although we do not study this bi-objective version of the problem in this work, we note that a solution with *minimum communication cost* implicitly requires a small team, since larger teams typically result in higher communication costs.

In our setting, we assume that individuals either have a skill or not; we do not allow for a scaling of the nodes' abilities. Similarly for the tasks; we assume that a task requires a certain set of skills, without considering the special importance that different skills might have for the completion of the task. Therefore, a straightforward generalization of the TEAM FORMATION problem would be its *graded* variant. In such a variant, the degree of skillfulness of individuals and the extent to which a skill is required for the completion of a task can be modelled by means of an integer weight in some interval, e.g., $\{0, 1, \ldots, \delta\}$. In this case, the task specification explicitly states for every required skill $a_j \in T$ the minimum level requirement δ_j. Similarly, for every individual i with skill a_j, the level of her competence with respect to a_j is specified. Then, all individuals with competence level higher or equal to the minimum required level are capable of contributing in covering this skill for the given task. Conceptually, we assume that an individual has a skill, only if his respective competence level is equal or higher to the required level. In this way, this "graded" version of the problem becomes identical to the basic version of the TEAM FORMATION problem, studied in this work.

6.5 Algorithms

In this section, we present algorithms for the DIAMETER-TF and MST-TF problems. Our algorithmic solutions exploit the relationship of these two problems with the MCC and GST problems, respectively.

6.5.1 Algorithms for the DIAMETER-TF problem

Algorithm 2 shows the pseudocode of the `RarestFirst` algorithm for the DIAMETER-TF problem. The algorithm is a variation of the *Multichoice* algorithm presented in [117]. First, for every skill a required by the task T, we compute $S(a)$, the support of a. Then, the algorithm picks the skill $a_{\text{rare}} \in T$ with the lowest-cardinality support $S\left(a_{\text{rare}}\right)$. Note that at least one individual from the set $S\left(a_{\text{rare}}\right)$ needs to be included in the solution. Among all candidates from the set $S\left(a_{\text{rare}}\right)$, the algorithm picks the one that leads to the smallest diameter subgraph, when connected to its closest individual in all other support groups $S\left(a\right)$ ($a \in T$ and $a \neq a_{\text{rare}}$).

Recall that in line 6 of Algorithm 2, $d\big(i, S\left(a\right)\big)$ is simply $\min_{i' \in S(a)} d(i, i')$. Also recall

Algorithm 2 The `RarestFirst` algorithm for the Diameter-Tf problem.

```
   Input: Graph $G(\mathcal{X}, E)$; individuals' skill vectors $\{X_1, \ldots, X_n\}$ and task $T$.
   Output: Team $\mathcal{X}' \subseteq \mathcal{X}$ and subgraph $G[\mathcal{X}']$.
1: for every $a \in T$ do
2:     $S(a) = \{i \mid a \in X_i\}$
3: $a_{rare} \leftarrow \arg\min_{a \in T} |S(a)|$
4: for every $i \in S(a_{rare})$ do
5:     for $a \in T$ and $a \neq a_{rare}$ do
6:         $R_{ia} \leftarrow d\big(i, S(a)\big)$
7:     $R_i \leftarrow \max_a R_{ia}$
8: $i^* \leftarrow \arg\min R_i$
9: $\mathcal{X}' = i^* \cup \{Path(i^*, S(a)) \mid a \in T\}$
```

that *Path* $(i^*, S(a))$ in line 9 is the set of nodes in the graph that are along the shortest path from i^* to i', where i' is such that $i' \in S(a)$ and $d\big(i^*, S(a)\big) = d(i^*, i')$. We assume that all pairs shortest path have been pre-computed, and we use hash tables for storing the attributes of every individual and a different set of hashtables for storing the individuals that posses a specific attribute. Then, the running time of the `RarestFirst` algorithm is $\mathcal{O}(|S(a_{rare})| \times n)$. A worst-case analysis suggests that $|S(a_{rare})| = \mathcal{O}(n)$. Thus the worst-case running time of the `RarestFirst` is $\mathcal{O}(n^2)$. However, in practice, the running time of the algorithm is much less that this worst-case analysis suggests.

Since the employed distance function d is a metric, we can state the following for the approximation factor of the `RarestFirst` algorithm:

Proposition 4. *For any graph-distance function d that satisfies the triangle inequality, the* Cc-R *cost of the solution $\mathcal{X}'$, given by* `RarestFirst` *for a given task, is at most twice the* Cc-R *cost of the optimal solution $\mathcal{X}^*$. That is,* Cc-R $(\mathcal{X}') \leq 2 \cdot$ Cc-R $(\mathcal{X}^*)$.

Proof. The analysis we present here is similar to the analysis of the *Multichoice* algorithm presented in [117]. First, consider the solution $\mathcal{X}'$ output by the `RarestFirst` algorithm, and let $a_{rare} \in T$ be the skill possessed by the least number of individuals in $\mathcal{X}$. Also, let i^* be the individual picked from set $S(a_{rare})$ to be included in the solution $\mathcal{X}'$. Now consider two other skills $a \neq a' \neq a_{rare}$ and individuals $i, i' \in \mathcal{X}'$ such that $i \in S(a), i \notin S(a')$ and

$i' \in S(a'), i' \notin S(a)$. If i, i' are part of the team reported by the `RarestFirst` algorithm, it means that $i = \arg\min_{j \in S(a)} d(i^*, j)$ and $i' = \arg\min_{j \in S(a')} d(i^*, j)$. Due to the way the algorithm operates, we can lowerbound the CC-R cost of the optimal solution as follows:

$$\text{CC-R}(\mathcal{X}^*) \geq d(i^*, i) \text{ and } \text{CC-R}(\mathcal{X}^*) \geq d(i^*, i'). \tag{6.1}$$

Since we have assumed that the distance function d satisfies the triangle inequality we also have that $d(i, i') \leq d(i^*, i) + d(i^*, i')$. By applying the bounds given in (6.1) in the triangle inequality, we get the proposed approximation factor.

$$\begin{aligned} d(i, i') &\leq d(i^*, i) + d(i^*, i') \\ &\leq \text{CC-R}(\mathcal{X}^*) + \text{CC-R}(\mathcal{X}^*) \\ &= 2 \cdot \text{CC-R}(\mathcal{X}^*). \end{aligned}$$

□

6.5.2 Algorithms for the MST-TF problem

In this section we describe two algorithms for solving the MST-TF problem: the `CoverSteiner` and `EnhancedSteiner` algorithms. Both algorithms are motivated by the resemblance of MST-TF to Steiner tree problems.

The `CoverSteiner` algorithm

Algorithm 3 The `CoverSteiner` algorithm for the MST-TF problem.

```
Input: Graph $G(\mathcal{X}, E)$; individuals' skill vectors $\{X_1, \ldots, X_n\}$ and task $T$.
Output: Team $\mathcal{X}' \subseteq \mathcal{X}$ and subgraph $G[\mathcal{X}']$.
$\mathcal{X}_0 \leftarrow$ GreedyCover($\mathcal{X}, T$)
$\mathcal{X}' \leftarrow$ SteinerTree($G, \mathcal{X}_0$)
```

The first heuristic we present for the MST-TF problem proceeds in two steps. In the first step, the social network is ignored and the algorithm focuses on finding a set of individuals $\mathcal{X}_0 \subseteq \mathcal{X}$ such that $\cup_{i \in \mathcal{X}_0} X_i \supseteq T$. In the second step, the algorithm finds the minimum cost tree that spans all the nodes in $\mathcal{X}_0$, and possibly other nodes in $\mathcal{X} \setminus \mathcal{X}_0$. In that way, a set of nodes $\mathcal{X}'$ such that $\mathcal{X}_0 \subseteq \mathcal{X}' \subseteq \mathcal{X}$ is reported. We call this two-step algorithm the `CoverSteiner` algorithm.

The pseudocode of this algorithm is given in Algorithm 3. The goal of the first step is to solve an instance of the classic SET COVER problem: the universe of elements to be covered are the requirements of task T and each individual in $\mathcal{X}$ is a subset of the universe. To solve this, we use the standard `GreedyCover` algorithm for the SET COVER problem. The `GreedyCover` algorithm is an iterative greedy procedure, adding at each step t the individual X_t that possesses the most yet uncovered required skills in T. For details on this algorithm see [121].

In its second step, the `CoverSteiner` algorithm solves an instance of the STEINER TREE problem on graph G. Recall that in the standard STEINER TREE problem, we are given an undirected graph with non-negative edge costs. The vertices of this graph are partitioned into two sets: the *required* and the *Steiner* vertices. The STEINER TREE problem then asks for the minimum-cost tree in the input graph that contains all required vertices and any subset of the Steiner vertices. In our case, the set of nodes $\mathcal{X}_0$ reported by the `GreedyCover` algorithm corresponds to the set of required vertices, while the vertices in $\mathcal{X} \setminus \mathcal{X}_0$ represent the Steiner vertices. Given graph $G(\mathcal{X}, E)$, the goal of line 2 of Algorithm 3 is to find the solution $\mathcal{X}'$ that minimizes CC-MST $(\mathcal{X}')$, under the constraint that $\mathcal{X}' \supseteq \mathcal{X}_0$.

There exist many algorithms for solving the classic STEINER TREE problem. The pseudocode of the algorithm we use for our experiments is given in Algorithm 4. We call this algorithm the `SteinerTree`. The algorithm is due to [122], and is in fact a greedy heuristic for the STEINER TREE. The algorithm incrementally adds to the current solution $\mathcal{X}'$ nodes from the required set $\mathcal{X}_0$. At every step, a single node from $\mathcal{X}_0$ is added; this is the

Algorithm 4 The `SteinerTree` algorithm.

```
    Input: Graph $G\left(\mathcal{X}, E\right)$; required nodes $\mathcal{X}_0$ and Steiner nodes $\mathcal{X} \setminus \mathcal{X}_0$.
    Output: Team $\mathcal{X}_0 \subseteq \mathcal{X}' \subseteq \mathcal{X}$ and subgraph $G\left[\mathcal{X}'\right]$.
1:  $\mathcal{X}' \leftarrow v$, where $v$ is a random node from $\mathcal{X}_0$.
2:  while $\left(\mathcal{X}_0 \setminus \mathcal{X}'\right) \neq \emptyset$ do
3:      $v* \leftarrow \operatorname{argmin}_{u \in \mathcal{X}_0 \setminus \mathcal{X}'} d\left(u, \mathcal{X}'\right)$
4:      if $\mathit{Path}\left(v^*, \mathcal{X}'\right) \neq \emptyset$ then
5:          $\mathcal{X}' \leftarrow \mathcal{X}' \cup \left\{\mathit{Path}\left(v^*, \mathcal{X}'\right)\right\}$
6:      else
7:          Return Failure
```

node v^* that has the minimum distance to the set of nodes $\mathcal{X}'$ already added to the solution (line 3). If such node exists v^* along with all the nodes in the shortest path from it to $\mathcal{X}'$ are added to the solution set. Otherwise, failure is reported.

The running time of the `CoverSteiner` algorithm is the summation of the running times of `GreedyCover` and `SteinerTree`. The time required for the execution of the `GreedyCover` algorithm is $\mathcal{O}\left(|T| \times |\mathcal{X}|\right)$ or $\mathcal{O}\left(mn\right)$. The time required for the execution of `SteinerTree` shown in Algorithm 4 is $\mathcal{O}\left(|\mathcal{X}_0| \times |E|\right)$. Thus, in the worst case, the running time of `CoverSteiner` is $\mathcal{O}\left(n^3\right)$ (this is because $|\mathcal{X}_0| = \mathcal{O}\left(n\right)$ and $|E| = \mathcal{O}\left(n^2\right)$). However, in practice the cardinalities of sets $\mathcal{X}_0$ and E are much less than their worst-case upper bounds.

The main disadvantage of the `CoverSteiner` algorithm is that, in the first step, it completely ignores the underlying graph structure. This can lead to teams with a high communication cost, or may even lead to failure, even in cases where a solution to the MST-TF problem actually exists.

The `EnhancedSteiner` algorithm

The inadequacies of the `CoverSteiner` algorithm can be alleviated by the `EnhancedSteiner` algorithm that we describe in this section.

The `EnhancedSteiner` algorithm starts by first enhancing graph G with additional nodes and edges to form the *enhanced graph* H. Then, `SteinerTree` is evoked to solve the Steiner Tree problem on the enhanced graph H (for similar applications of Steiner tree algorithms see [123]). The pseudocode that corresponds to these two steps of the `EnhancedSteiner` algorithm is shown in Algorithm 5.

Algorithm 5 The `EnhancedSteiner` algorithm for the Mst-Tf problem.

```
   **Input:** Graph $G(\mathcal{X}, E)$; individuals' skill vectors $\{X_1, \ldots, X_n\}$ and task $T$.
   **Output:** Team $\mathcal{X}' \subseteq \mathcal{X}$ and subgraph $G[\mathcal{X}']$.
1: $H \leftarrow$ EnhanceGraph$(G, T)$
2: $\mathcal{X}_H \leftarrow$ SteinerTree$(H, \{Y_1, \ldots, Y_k\})$
3: $\mathcal{X}' \leftarrow \mathcal{X}_H \setminus \{Y_1, \ldots, Y_k\}$
```

Let the task to be performed require k skills, i.e., $T = \{a_1, \ldots, a_k\}$. The routine `Enhance` (line 1 of Algorithm 5) makes a linear pass over the graph G and enhances it as follows: an additional node Y_j is created for every skill $a_j \in T$. Each such new vertex Y_j is connected to a node $i \in \mathcal{X}$ if and only if $a_j \in X_i$. The distance between node Y_j and nodes $i \in S(a_j)$ are set to be $d(Y_j, i) = D$ where D is a large real number, larger than the sum of all the pairwise distances of the nodes in the graph G. Finally, every node $i \in \mathcal{X}$ that has abilities X_i is replaced by a clique C_i of size $|X_i|$. Each node in the clique C_i should be considered as a copy of individual i that has only a single distinct skill from the set X_i. The distance between every two nodes in the clique C_i is set to zero. Each node in the clique C_i maintains all the existing connections of node i to the rest of the graph – including the connections to nodes $\{Y_1, \ldots, Y_k\}$.

The set of nodes $\mathcal{X}_H$ that participate in the Steiner tree of the enhanced graph H are found by calling the `SteinerTree` algorithm with required nodes $Y_1, \ldots, Y_k$. In a final step, the algorithm removes from set $\mathcal{X}_H$ the artificially added nodes $Y_1, \ldots, Y_k$ (and their incident edges) to obtain the final solution $\mathcal{X}'$.

The following claim can be made with respect to this algorithm. Let $\mathcal{X}_H^*$ be the set of nodes in the *optimal Steiner tree* of the enhanced graph H, and $\mathcal{X}^*$ be the optimal team for the Mst-Tf problem. Then, we have that Cc-Mst $(\mathcal{X}^*) =$ Cc-Mst $(\mathcal{X}_H^* \setminus \{Y_1, \ldots, Y_k\})$. That is, if we remove nodes $Y_1, \ldots, Y_k$ (and their incident edges) from the optimal solution of the Steiner tree problem on the enhanced graph H, then the remaining nodes form the optimal solution to the MST-TF problem.

Observe that the replacement of every individual i with a clique C_i of size $|X_i|$ is only conceptual. In practice, the implementation of the algorithm does not require this. Therefore, the enhanced graph H contains only k more nodes than the input graph G, namely the nodes $Y_1, \ldots, Y_k$. Therefore, following the analysis of the `SteinerTree` done in the previous section, we have that the running time of the `EnhancedSteiner` algorithm is $\mathcal{O}\left(k \times |E|\right)$.

The `EnhancedSteiner` algorithm is motivated by the obvious similarity between the MST-TF problem and the GROUP STEINER TREE (GST) problem; the connection was already highlighted in the proof of Proposition 3. In general, instead of the `EnhancedSteiner` algorithm, any other (approximation) algorithm for the GST problem can also be used to solve the MST-TF problem. We have picked the `EnhancedSteiner` algorithm because it is simple, intuitive and works well in practice. The best approximation ratio achieved by an algorithm is $O(\log^3 n \log k)$ [124]. For a review of some recent approximation algorithms for the GST problem see [119, 123, 124] and references therein.

6.6 Experimental evaluation

In this section we evaluate the proposed algorithms for the TEAM FORMATION problem using the scientific-collaboration graph extracted from the DBLP bibliography server. We show that our algorithms for both the DIAMETER-TF and MST-TF problems give high-quality results in terms of the *communication cost*, *the cardinality of the team*, and *the connectivity of the team*. Examples of teams reported by our methods illustrate the effectiveness of our framework in real scenarios.

6.6.1 Other algorithms

In addition to the algorithms we described in Section 6.5, we also experiment with some straightforward greedy heuristics, that would be natural alternatives for solving the TEAM FORMATION problem. The rationale of these algorithms is to form a solution iteratively. At round t, team $\mathcal{X}_t$ is formed by adding to the team $\mathcal{X}_{t-1}$ a node $i \in \mathcal{X} \setminus \mathcal{X}_{t-1}$. The node i is selected so that it maximizes the ratio

$$i = \operatorname*{argmax}_{i' \in \mathcal{X} \setminus \mathcal{X}_{t-1}} \frac{\left| C\big(\mathcal{X}_{t-1} \cup \textit{Path}\left(\mathcal{X}_{t-1}, i'\right), T\big) - C\big(\mathcal{X}_{t-1}, T\big) \right|}{\textsc{Cc}\big(\mathcal{X}_{t-1} \cup \textit{Path}\left(\mathcal{X}_{t-1}, i'\right)\big)}.$$

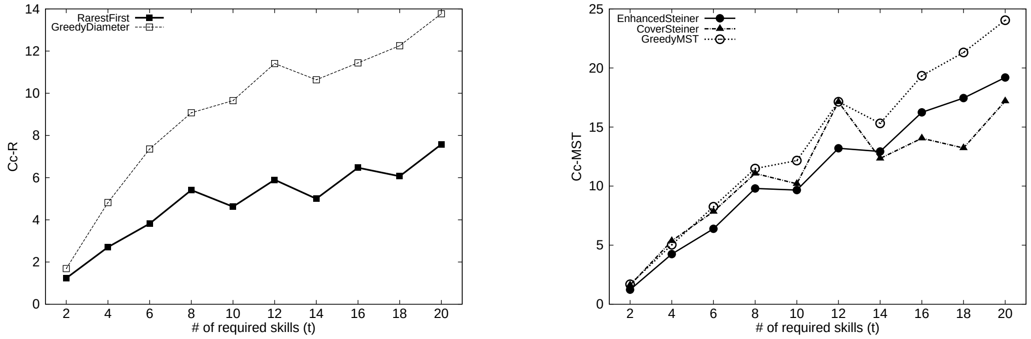
That is, the node i that achieves the best ratio of newly covered skills in T divided by the corresponding communication cost is picked. We refer to the variation of the greedy algorithm that uses the CC-R (resp. CC-MST) communication-cost function, as `GreedyDiameter` (resp. `GreedyMST`).

6.6.2 The DBLP dataset

We use a snapshot of the DBLP data taken on April 12, 2006 to create a benchmark dataset for our experiments. We only keep entries of the snapshot that correspond to papers published in the areas of *Database* (DB), *Data mining* (DM), *Artificial intelligence* (AI) and *Theory* (T) conferences. For each paper, we have information about its authors (names), title, the forum where it was published and the year of publication. We end up with a total of 19 venues categorized as follows: DB $=$ {SIGMOD, VLDB, ICDE, ICDT, EDBT, PODS}, DM $=$ {WWW, KDD, SDM, PKDD, ICDM}, AI $=$ {ICML, ECML, COLT, UAI} and T $=$ {SODA, FOCS, STOC, STACS}. We refer to the set of selected papers as the **DBLP** dataset.

We now proceed to generate the input to the TEAM FORMATION Problem as follows. The set of skilled individuals $\mathcal{X}_{\text{dblp}}$ consists of the set of authors that have at least three papers in the **DBLP** dataset. The skillset X_i of each such author i consists of the set of terms that appear in *at least two titles* of papers in **DBLP** that he has co-authored. The above procedure creates a set $\mathcal{X}_{\text{authors}}$ consisting of 5508 individuals and 1792 distinct skills. Two authors i, i' are connected in the graph G_{dblp} $(\mathcal{X}_{\text{dblp}}, E)$ if they appear as co-authors in *at least two papers* in **DBLP**. This threshold leads to a graph G_{dblp} that has 5588 total edges. The weight of an edge connecting nodes i, i' is $w(i, i') = 1 - \frac{|P_i \cap P_{i'}|}{|P_i \cup P_{i'}|}$; P_i (resp., $P_{i'}$) is the set of papers authored by i (resp., i'). In other words, the weights on the edges represent pairwise Jaccard distances between all pairs of connected nodes. We compute the graph distance between two nodes in graph G_{dblp} using the shortest path distance as we described in Section 6.3.

6.6.3 Performance Evaluation



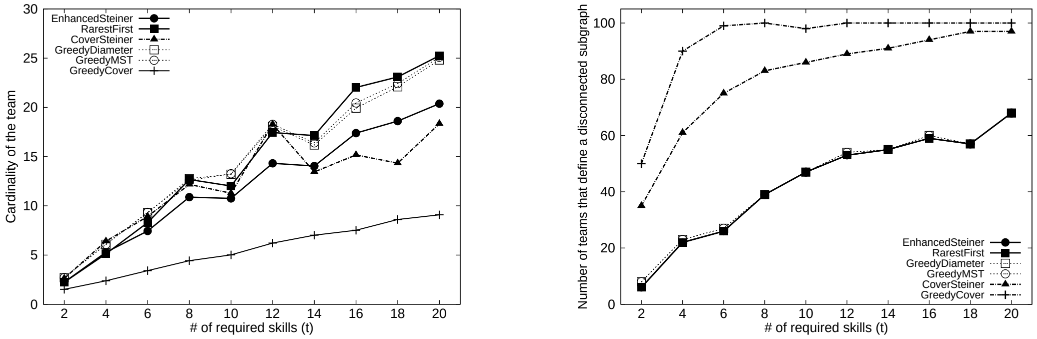
(a) CC-R cost

(b) CC-MST cost

Figure 6.2: Average communication cost of the teams produced by each TEAM FORMATION algorithm for tasks $T(t, 1)$ with $t \in \{2, 4, \ldots, 20\}$. Figure 6.2(a): Average CC-R cost of `RarestFirst` and `GreedyDiameter` algorithms. Figure 6.2(b): Average CC-MST cost of `EnhancedSteiner`, `CoverSteiner` and `GreedyMST` algorithms.

This section evaluates the TEAM FORMATION algorithms on the *communication cost*, the *cardinality of the team* and the *connectivity of the team*.

Task generation: Every generated task is characterized by two parameters: 1) t – the number of required skills in the task; and 2) s – the diversity of the required skills in terms of their corresponding areas. We use $T(t, s)$ to refer to a task generated for a specific



(a) Cardinality of the team.

(b) Number of disconnected teams.

Figure 6.3: Figure 6.3(a): Average cardinality of the teams reported by `RarestFirst`, `EnhancedSteiner`, `CoverSteiner`, `GreedyDiameter`, `GreedyMST`, `GreedyCover`. Figure 6.3(b): Number of reported teams that define a subgraph with disconnected components. The count is taken over 100 independent tasks generated for every $T(t,1)$ where $t \in \{2,4,\ldots,20\}$.

configuration of these parameters.

Specifically, a task $T(t,s)$ is generated as follows: first, we select a subset of the research areas $S \subseteq \{$ DB, DM, AI ,T $\}$ with $|S| = s$. Then, we randomly pick t required skills from the terms appearing in papers published in conferences belonging to these areas. For the results we report in this section we use $t \in \{2,4,\ldots,20\}$ and $s = 1$. For every (s,t) configuration we generate 100 random tasks for this configuration and report the average results obtained by the different methods. Experiments for $s = 2,3,4$ exhibit similar trends as those for $s = 1$ and thus are not presented due to space constraints.

Communication cost: Figure 6.2(a) shows the average CC-R costs of the solutions achieved by `RarestFirst` and `GreedyDiameter` on tasks $T(t,1)$ with $t \in \{2,4,\ldots,20\}$. Figure 6.2(b) shows the average CC-MST costs of the `EnhancedSteiner`, `CoverSteiner` and `GreedyMST` algorithms on the same set of tasks. Note that the average is calculated for the solutions $\mathcal{X}'$ that result in a connected graph $G[\mathcal{X}']$. If, for a specific task, the solution produced by a specific algorithm does not lead to a connected graph, we simply ignore it.

It can be observed that, in terms of the diameter cost, `RarestFirst` significantly outperforms `GreedyDiameter`. Similarly, in terms of the MST cost, `EnhancedSteiner` generally gives better results than `CoverSteiner` and `GreedyMST`. The conclusion is that our proposed algorithms can form teams that are able to accomplish a given task with low communication efforts.

Cardinality of the team: Since the size of the team often has a positive correlation with the expenses of a project, we evaluate the cardinality of the teams formed by every TEAM FORMATION algorithm. The results in Figure 6.3(a) show that the `RarestFirst` algorithm tends to report relatively large teams, especially for large values of t. On the other hand, the `EnhancedSteiner` algorithm generally finds teams of small size. This can be explained by the fact that the `RarestFirst` algorithm aims to minimize the diameter of the graph, which is less likely to be affected by the introduction of new nodes. On the other hand, the `EnhancedSteiner` algorithm tries to minimize the MST cost, which is always increased when a new node is added to the team.

For comparison purposes, we also include the cardinality of the teams reported by the `GreedyCover` algorithm. Recall that `GreedyCover` ignores the existence of the graph and only reports a set of individuals who can perform the task by simply looking at their skillsets. Therefore, the cardinality of this solution is a lower bound on the cardinality of the solutions produced by all the five aforementioned algorithms. However, since `GreedyCover` ignores the graph structure, it often forms teams of members that cannot communicate. That is, the subgraph of the original graph defined by the members of such teams is not connected. The following experiment illustrates the validity of this claim.

Connectivity of the team: Given a task T, it might be the case that there does not exist a team $\mathcal{X}'$ such that the members of $\mathcal{X}'$ simultaneously have all the skills required by T, and also define a connected subgraph. Further, even if such a team exists, it might be the case that some algorithms fail to find it. In this experiment, we evaluate the effectiveness of the different algorithms in finding teams that correspond to connected subgraphs of the original graph. Recall that connected subgraphs have significantly lower communication costs (both CC-R and CC-MST) than disconnected ones.

Figure 6.3(b) shows, for every algorithm and every $t \in \{2, 4, \ldots, 20\}$, the number of times a team formed by an algorithm defines a disconnected subgraph. The count is taken over the 100 independent tasks generated for every $T(t, 1)$. We can observe that `RarestFirst`, `GreedyDiameter`, `EnhancedSteiner` and `GreedyMST` produce approximately the same number of disconnected teams. We conjecture that the tasks for which these algorithms fail to report a connected subgraph are in fact those that have no connected team as a solution. On the other hand, `CoverSteiner` and `GreedyCover` often fail to find a connected team, even in cases where such a team actually exists. The results indicate that, although `GreedyCover` produces teams of small size, the members of this team cannot communicate efficiently.

6.6.4 Qualitative evidence

The goal of this experiment is to show that our problem definitions and their corresponding algorithms produce reasonable and intuitive results in practical settings. As input to our problem, we again consider the individual authors in $\mathcal{X}_{\text{dblp}}$ and the corresponding co-authorship graph G_{dblp}, that we described in Section 6.6.2. We test our framework on 10 distinct tasks. The required skills for each task are defined by the words appearing in the title of an already published paper. The papers were chosen from the "Most Cited Computer Science Articles" list, maintained by CiteSeerX (`citeseerx.ist.psu.edu/stats/articles`). We thus form 10 tasks by selecting the top-10 cited papers from the list, which were also published in one of the 19 conferences covered by the **DBLP** Dataset. Table 6.1 shows the titles of the these papers.

Table 6.2 shows the ten teams of authors obtained by the `RarestFirst` and `EnhancedSteiner` algorithms. The set of original authors for every paper is also reported. The names highlighted in bold in the last two columns of the table indicate authors that have been selected because they covered some required skill of the input task. The names appearing not in bold correspond to authors that were included in the team as mediators, i.e., communication nodes that ensure the connectivity of the graph.

Table 6.1: Titles of the top-10 most cited papers from the **DBLP** dataset according to CiteSeerX citation counting. The keywords appearing in the tiles define the required skills of 10 distinct tasks.

Rank	Paper title
1	The anatomy of a large-scale hypertextual Web search engine
2	Fast algorithms for mining association rules
3	Mining association rules between sets of items in large databases
4	Text categorization with support vector machines: Learning with many relevant features
5	Conditional random fields: Probabilistic models for segmenting and labeling sequence data
6	Mining frequent patterns without candidate generation
7	A survey of approaches to automatic schema matching
8	Automatic subspace clustering of high dimensional data for data mining applications
9	Models and issues in data stream systems
10	NiagaraCQ: A Scalable Continuous Query System for Internet Databases

We can observe that for papers 3, 6, and 9, `RarestFirst` finds a single-node solution, whereas `EnhancedSteiner` fails to do so. This is due to the fact that `EnhancedSteiner` starts with a random node from $\mathcal{X}_0$, so it may be the case that none of the nodes in the final team possesses all the required skills. On the other hand, `RarestFirst` examines every node who has the skill with the lowest-cardinality support. If a node of them happens to have all other required skills, the process simply reports that node and terminates.

In general, both algorithms produce teams of reasonable size; note that not too many mediator nodes (nodes without skill contribution) are introduced. In many cases, the actual authors of a paper were included in the formed team. This is reasonable, since the real teams are more likely to combine skill coverage with a low communication cost. This attests not only to the effectiveness of the algorithms, but also to the validity of the problem definitions.

6.7 Conclusions

In this work, we addressed the problem of forming a team of skilled individuals to perform a given task, while minimizing the communication cost among the members of the team. We explored two alternative formulations for the communication cost, which we believe are practical and intuitive. We proved that the TEAM FORMATION problem is NP-Hard for both formulations and proposed appropriate approximation algorithms. In a thorough experimental evaluation, we evaluated the performance of our algorithms, and compared them against reasonable baseline approaches. We concluded with a qualitative evaluation, reporting the teams formed by our algorithms on a set of real tasks.

Table 6.2: Authors of the top-10 most cited papers from the **DBLP** dataset; **column 1**: paper ranking in the top-10 list; **column 2**: the actual authors of the papers; **column 3**: authors suggested by the `RarestFirst` algorithm; **column 4**: authors suggested by the `EnhancedSteiner` algorithm

Rank	Actual authors	`RarestFirst` result	`EnhancedSteiner` result
1	S. Brin, L. Page	**Paolo Ferragina, Patrick Valduriez, H. V. Jagadish, Alon Y. Levy, Daniela Florescu** Divesh Srivastava, S. Muthukrishnan	**P. Ferragina ,J. Han, H. V. Jagadish, Kevin Chen-Chuan Chang, A. Gulli**, S. Muthukrishnan, Laks V. S. Lakshmanan
2	R. Agrawal, R. Srikant	**R. Agrawal**	**Philip S. Yu**
3	R. Agrawal, T. Imielinski, A. N. Swami	**Philip S. Yu**	**Wei Wang, Philip S. Yu**
4	T. Joachims	**Wei-Ying Ma, Gui-Rong Xue, H. Liu, J. Han, H. Lu, Z. Chen**, Q.Yang, H. Cheng	**J. Han, H. Lu, Wei-Ying Ma, Z. Chen, H. Liu, Gui-Rong Xue**, Q. Yang
5	J. Lafferty, F. Pereira, A. McCallum	**A. McCallum**	**A. McCallum**
6	J. Han, J. Pei, Y. Yin	**F. Bonchi**	**A. Gionis, H. Mannila, R. Motwani**
7	E. Rahm, P. A. Bernstein	**C. Bettini, R. Agrawal, Kevin Chen-Chuan Chang**, T. Imielinski, H. Garcia-Molina, D. Barbara, S. Jajodia	**C. Bettini, P. A. Bernstein**, H. Garcia-Molina, S. Jajodia, D. Maier, D. Barbara
8	R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan	**D. Gunopulos, R. Agrawal**	**R. Agrawal, D. Gunopulos**
9	B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom	**M. T. Ozsu**	**H. V. Jagadish, D. Srivastava**
10	J. Chen, D. J. DeWitt, F. Tian, Y. Wang	**Donald Kossmann, David J. DeWitt, Michael J. Franklin, Michael J. Carey**	**M. J. Carey, M. J. Franklin, D. Kossmann, D. J. DeWitt**

Chapter 7

Finding Effectors in Social Networks

7.1 Introduction

Consider the directed network shown in Figure 7.1, where the black nodes are *active* and the white nodes are *inactive*. The activation state of the network is described by an *activation vector*, $\mathbf{a}$. In the example of Figure 7.1, $\mathbf{a}(x) = \mathbf{a}(y_i) = 1$ for $0 \leq i \leq 2$ and $\mathbf{a}(x_i) = 0$ for $1 \leq i \leq \ell$. Assume a simple probabilistic information-propagation model such that every node v that becomes active activates a neighbor x via a directed link $(v \rightarrow x)$; this activation succeeds with probability equal to the weight of the directed link $(v \rightarrow x)$. Given a budget k, our goal is to find a set of k active nodes, such that, had the propagation started from them, it would have caused an activation state similar to the one described by $\mathbf{a}$. We call these nodes *effectors*[1] and the corresponding optimization problem the k-EFFECTORS problem. Effectors need not be the nodes that first became active during the information-propagation process; therefore, complete knowledge of the timestamps associated with the activation of every node would not necessarily help in identifying the effectors. Further, effectors need not be centrally-located in the network. They are simply the nodes that best explain the observed activation vector.

[1] In biochemistry, an effector is a substance that increases or decreases the activity of an enzyme.

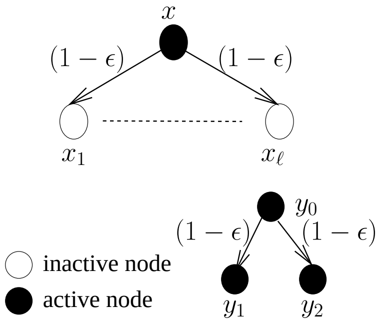


Figure 7.1: A network with active (black) and inactive (white) nodes. Edge weights represent the probability of an active node activating its neighbors; $\epsilon \in (0, 1)$.

In our example, assume that $k = 2$, $0 < \epsilon < 1$ and let the set of effectors be $X = \{x, y_1\}$. For this set X and the given propagation model, the *expected* final state of the propagation process assigns to every node v a probability of being active $\alpha(X, v)$. In this example, $\alpha(X, x) = \alpha(X, y_1) = 1$, $\alpha(X, x_i) = (1-\epsilon)$ for $1 \leq i \leq \ell$, $\alpha(X, y_0) = 0$ and $\alpha(X, y_2) = 0$. We define the cost of solution X to be $C(X) = \sum_{v \in V} |\mathbf{a}(v) - \alpha(X, v)| = (1-\epsilon)\ell + 2$. On the other hand, solution $X' = \{y_0, y_1\}$ would have cost $C(X') = 1 + \epsilon$ and it would be the optimal solution for every $\epsilon \in (0, 1)$.

In *social networks*, the identification of effectors can improve our understanding of the dynamics of information propagation. Effectors can be interpreted as key nodes that determine whether a novel concept dies out quickly or propagates to cover a significant portion of the network. In *epidemiological studies*, the effectors are the key individuals (or countries) that cause a particular diffusion pattern. The discovery of effectors can be leveraged in the design of vaccination strategies and quarantine policies. In *computer networks*, the effectors are computers in the network that affect the spread pattern of a computer virus. Again, effector discovery can facilitate inoculation strategies: rather than blindly investing on security software for protecting large parts of the network, system administrators can only focus on securing the effector nodes.

Our contribution: In this work, we first introduce the k-EFFECTORS problem and explore its connections to other existing problems in the literature. We prove that, in a general setting, the k-EFFECTORS problem is not only NP-hard to solve optimally, but also NP-hard to approximate. We also show that, in trees, the k-EFFECTORS problem can be solved optimally in polynomial time by using an efficient dynamic-programming algorithm. We also explore the performance of other computationally-efficient heuristics. Although our worst-case analysis shows that these heuristics are clearly suboptimal, our experimental evaluation reveals that, in certain settings, they can perform reasonably well. Finally, we experimentally validate our methods on the co-authorship graph defined by the **DBLP** dataset. More specifically, we use the **DBLP** co-authorship graph to find effectors of topics that appear in computer-science papers. We present qualitative evidence to show that the effectors identified by our methods convey meaningful information about the data.

We believe that the notion of effectors can improve our understanding of diffusion processes in networks. Although we focus our attention on the role of effectors in social networks, our framework can be applied to a variety of network data – including computer and biological networks – and give useful insights to the data analysts.

Our approach: Our approach for solving the k-EFFECTORS problem on tree networks consists of an optimal dynamic-programming algorithm. For general graphs, we proceed in two steps: first, for a given network and activation vector, we construct the *most probable tree* $\mathcal{T}$, that spans all the active nodes in the network. Then, we use the optimal dynamic-programming algorithm to identify the optimal effectors on $\mathcal{T}$. We believe that the extraction of the most probable tree from the input graph is interesting in its own right, since this tree models the backbone of information propagation in the network.

Roadmap: The rest of the work is organized as follows: in Section 7.2 we survey the related work and in Section 7.3 we give the necessary notation and describe the information-propagation model. The problem definition and the complexity results are presented in Section 7.4. In Section 7.5 we describe the optimal polynomial-time algorithm for trees and in Section 7.6 we present our algorithm for extracting the most probable tree from any given input graph. In Section 7.7 we provide a thorough set of experiments on on a co-authorship graph and we conclude in Section 7.8.

7.2 Related work

To the best of our knowledge, we are the first to formally define and study the k-EFFECTORS problem. Although the exact combinatorial definition of effectors does not exist in the literature, there has been a lot of work on problems related to the identification of *influential* nodes or trends in social or other networks. As expected, different definitions of influential nodes lead to different computational challenges. We summarize some of this work here.

In the blogosphere, there is significant research in the identification of influential blogs [125] and bloggers [126, 127]. Similarly, for marketing surveys, the problem of identifying the set of early buyers has been addressed [128]. However, the algorithmic settings are very different from ours. For example, Gruhl et. al. [125] study information diffusion of various topics in the blogoshere. The focus is on studying how the topics propagate or how "sticky" the topics are. In our setting, we do not touch upon the issue of durability of the trends; once a node becomes active, it remains active. In other studies, the focus is on the identification of influential bloggers [126, 127]. In these cases, the authors define a metric that determines the influence potential of a blogger. The focus is on developing efficient algorithms for computing the top-k influential nodes. In contrast, we evaluate groups of effectors and how they collectively affect the network.

Further, the aforementioned papers do not explicitly take into account the information-propagation model. Information-propagation models have been considered in the context of influence maximization [129, 130, 131]. The focus of those works is on identifying the set of nodes in the network that need to be targeted (e.g., for targeted advertisement), so that the propagation of a product or an idea spreads as much as possible. In influence maximization, the goal is to identify the nodes that will cause the most propagation effect in the network. In our case, the goal is to identify the nodes that better explain a particular - *ob-*

served- activation pattern in the network. In fact, our problem definition contains influence maximization as a special case.

Bharathi et al. [129] consider the influence-maximization problem on bidirectional trees and develop an FPTAS. In addition to the fact that their objective function is different from ours, they also focus on *undirected* trees. Our own focus is on directed graphs and directed trees. For undirected trees, their problem is NP-complete. On the other hand, ours is solvable in polynomial time for the case of directed trees.

The problem of identifying early adopters from transaction data has been addressed by Rusmevichientong et. al. [128]. In that work, the set of early buyers is identified by taking as input the detailed purchase information of each consumer. Then, a weighted directed graph is constructed: the nodes correspond to consumers and the edges to purchases these consumers have in common. Identifying early buyers corresponds to the problem of finding a subset of nodes in the graph with maximum difference between the weights of the outgoing and incoming edges. Contrary to our setting, the framework proposed by Rusmevichientong et. al. does not consider any information-propagation dynamics or any underlying social network.

The problem of finding links and initiators was also studied by Mannila and Terzi [132]. Their problem-setting is the following: given a set of individuals and the set of items each of them has purchased, the goal is twofold: a) For each item, identify the individuals that acted as its initiators. b) Infer the social relationships between individuals. The main difference between that work and our current work, is that here we assume that the social graph is given as part of the input. Further, we identify the set of effectors while ignoring the temporal information associated with the purchased items. Finally, the method of Mannila and Terzi is based on an MCMC sampling of the space of all possible graphs and initiators. Here, we solve the optimization problem of finding the *best* set of effectors rather than assigning probabilities to nodes being effectors.

Other definitions of “important” nodes in a network focus on the development of network inoculation strategies [133] and early epidemic detection [134, 135]. Since our goal is to find a set of effectors that best explain the network's final state, both our problem definition as well as our algorithmic approaches are significantly different.

7.3 Preliminaries

We assume a social network represented by graph a $G = (V, E, p)$. The nodes in V correspond to individuals. There is an edge between two individuals $u, v \in V$ if u and v are associated with each other. The edges in the network are *directed*; edge $(u \rightarrow v) \in E$ is associated with an *influence weight* $p(u \rightarrow v) \in [0, 1]$. This weight quantifies the effect that node u has on the decisions of node v. We give a probability interpretation to this weight. Note that we use the terms "graph", "social network" and "influence graph" interchangeably.

We assume that the influence weights are part of the input. For example, one can ask the users themselves to assign their own estimates of how much they are influenced by their own friends. Alternatively, one can employ a machine-learning algorithm to infer such probabilities [136]. For our experiments, we use a simple and intuitive method for computing the influence probabilities. The details of this computation are given in Section 7.7. The exploration of alternative methods for such computation, though interesting, is beyond the scope of this work.

Further, we assume that the influence of one node to another is the same for all items that propagate in the network. Exploring the performance of more specialized techniques that cluster the items and compute different influence probabilities per cluster is beyond the scope of this work.

Apart from the network and the influence probabilities, we also assume a particular (information) item I. For every node $v \in V$, an item I either appears or does not appear in v. We represent this information using a 0–1 $n \times 1$ vector $\mathbf{a}$; $\mathbf{a}(i) = 1$ if item I is observed at node i. Otherwise, $\mathbf{a}(i) = 0$. If $\mathbf{a}(i) = 1$ (resp. $\mathbf{a}(i) = 0$) we say that node i is *active* (resp. *inactive*). We call this vector the *activation vector* of I. Note that we assume that the entries of the activation vector are either 0 or 1. However, all our results carry over to the case where the observed activation vector takes real values in the interval $[0, 1]$.

7.3.1 The information-propagation model

We consider the following information-propagation model in a social network: when node u becomes active for the first time at step t, it gets a single chance to activate node v through the edge $(u \rightarrow v)$; u succeeds in this activation attempt with probability $p(u \rightarrow v)$ – as defined in the influence graph. If u succeeds, then v will become active at step $(t + 1)$. Otherwise, u cannot make any more attempts to activate v in any subsequent rounds. This model is called the *Independent Cascade* (IC) model [137, 138, 131]. IC is a *probabilistic* propagation model, since the activation process is influenced by probabilistic choices. Given a seed of nodes that are originally active, each node in the network is active with some probability. upon the termination of the process.

In the special case where all the the influence weights are equal to one, the IC model becomes equivalent to the *deterministic propagation* (DM) model. In the DM model every node that becomes active at step t activates all its neighbors with probability 1. Therefore, the activation of a single node in a strongly-connected component [2] is sufficient to activate all the nodes in the component.

Although we focus our attention on the IC model, our framework can be combined with any information-propagation model, including the *Linear Threshold* (LT) model [131] or the *Susceptible - Infected - Susceptible* (SIS) model [134].

Given a set $X \subseteq V$ of originally active nodes, the propagation of information with IC will terminate in at most n discrete timestamps. Since the information-propagation model

[2] In a strongly-connected component of a directed graph there is a directed path from every node to every other node of the component.

is non-deterministic, one needs to compute the probability that a node $v \in V$ is active at the end of the process. Computing this probability, denoted by $\alpha(v, X)$, requires exponential time in arbitrary graphs. On the other hand, one can estimate it by using the following simple heuristic: For graph $G = (V, E, p)$ keep every edge $(u \to v)$ with probability $p(u \to v)$. The edges of the resulting graph G' have influence probabilities equal to 1. That is, one can run the DM model on G'. After repeating this process N times, one can estimate $\alpha(v, X)$ by simply counting the fraction of the times v was active in the sampled graphs.

Further, if $G = (V, E, p)$ is a *directed* tree, then for $X \subseteq V$, we can find a closed-form expression of $\alpha(v, X)$. That is, for every node v we have that:

$$\alpha(v, X) = 1 - \prod_{x \in X} \left(1 - \prod_{(y \to z) \in \text{path}(x,v)} p(y \to z) \right). \tag{7.1}$$

The term inside the parenthesis corresponds to the probability that node v does not get influenced by node x. Therefore, the outer product computes the probability that node v does not get active. The probability that v gets active is, naturally, one minus this product.

7.4 The Problem

Assuming a particular information-propagation model, our goal is to solve the following problem.

Problem 9 (k-EFFECTORS problem)**.** *Given a social network graph $G = (V, E, p)$ and an activation vector $\mathbf{a}$, find a set X of active nodes (effectors), of cardinality at most k such that*

$$C(X) = \sum_{v \in V} |\mathbf{a}(v) - \alpha(v, X)| \tag{7.2}$$

is minimized.

The k-EFFECTORS problem asks for the set of individuals that, once activated, cause an activation pattern which is as similar as possible to the activation observed in vector $\mathbf{a}$. We also use $C(v, X)$ to refer to the contribution of node v in the cost function. In other words, we define $C(v, X) = |\mathbf{a}(v) - \alpha(v, X)|$ and thus $C(X) = \sum_{v \in V} C(v, X)$.

The definition of the k-EFFECTORS problem is independent of the information-propagation model. Although some of our results generalize to many information-propagation models, we focus here on the IC model. Also, we restrict the effectors to be selected from the set of active nodes. Although allowing any node (active or inactive) to be an effector would not change our theoretical results, we put this constraint mostly because picking inactive nodes as effectors contradicts our intuition.

Next, we study the complexity of the k-EFFECTORS problem under the IC propagation model. For the complexity results we use the decision version of the k-EFFECTORS problem, which we parameterized by cost c. That is, k-EFFECTORS(c) is formulated as the following decision problem: Given a social network $G = (V, E, p)$ and an activation vector $\mathbf{a}$ does there exist a set $X \subseteq V$, $|X| \leq k$ with $C(X) \leq c$? We begin by proving the following lemma.

Lemma 2. *Assuming the IC propagation model, the* k-EFFECTORS*(0) problem is NP-complete.*

Proof. Consider an instance of the NP-complete SET COVER problem, defined by a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of a ground set $U = \{u_1, u_2, \ldots, u_n\}$. The question is whether there exist k subsets from $\mathcal{S}$ whose union is equal to U. Given an arbitrary instance of the SET COVER PROBLEM, we define the corresponding graph G to be a directed graph with $n + m + 1$ nodes. There is a node i corresponding to each set S_i, a node j corresponding to each element u_j, and a directed edge $(i \to j)$ with influence probability $p(i \to j) = 1$ whenever $u_j \in S_i$. The $(n + m + 1)$-th node of G is node ℓ. Every node

j (corresponding to element u_j) is connected to node ℓ via a directed edge with weight $p(j \rightarrow \ell) = 1/n$. Finally, node ℓ is connected every node i (that correspond to set S_i) via a directed edge with probability $p(\ell \rightarrow i) = 1$. Finally, we set the activation vector so that all nodes in the graph G are active, i.e., $\mathbf{a} = \vec{1}$. There exists a solution consisting of k sets to the SET COVER problem if and only if there exists a set X of k effectors in this graph with cost $C(X) = 0$. The problem is trivially in NP. □

Lemma 2 allows us to prove the following inapproximability result.

Lemma 3. *Assuming the IC propagation model, there does not exist a β-approximation algorithm for the k-*EFFECTORS *problem, with $\beta > 1$, unless P $=$ NP.*

Proof. The proof is by contradiction. Assume that there is a polynomial time β-approximation algorithm for the k-EFFECTORS problem; call this algorithm `Approx`. For any instance $G = (V, E, p)$ and activation vector $\mathbf{a}$, `Approx` will produce a solution $X \subseteq V$ such that $C(X) \leq \beta C(X^*)$, where X^* is the optimal solution. Assume now an instance of the k-EFFECTORS(0) problem (see Lemma 2). If we give this instance as input to the `Approx` algorithm then, `Approx` should be able to *decide* whether there is a 0-cost solution to the instance or not. However, from Lemma 2, we know that k-EFFECTORS(0) is NP-complete and thus we reach a contradiction. □

In fact, the k-EFFECTORS problem is a generalization of the INFLUENCE MAXIMIZATION problem [131]. In our context, the INFLUENCE MAXIMIZATION problem asks for the set $Y \subseteq V$ with $|Y| \leq k$, such that $\sum_{v \in V} \alpha(v, Y)$ is maximized. Maximizing $\sum_{v \in V} \alpha(v, Y)$ is equivalent to minimizing $\sum_{v \in V}(1 - \alpha(v, Y))$. Thus, when the activation vector $\mathbf{a}$ contains all 1's, i.e., $\mathbf{a} = \mathbf{1}$, the two problems are equivalent.

This observation allows us to infer that the k-EFFECTORS problem is NP-complete for all the information propagation models used by Kempe et al. [131]. In fact, by the results of Kempe et. al. [131] (due the construction used in the proof of Theorem 2.4), we also know that INFLUENCE MAXIMIZATION, for the IC propagation model, is NP-complete even for Directed Acyclic Graphs (DAGs). As a result the k-EFFECTORS problem is also NP-complete for DAGs.

Corollary 1. *Assuming the IC propagation model, the* k-EFFECTORS *problem is NP-complete even when the input graph* $G = (V, E, p)$ *is a DAG.*

However, for the DM propagation model, the k-EFFECTORS problem can be solved optimally in polynomial time. The polynomial-time algorithm first finds all the strongly connected components of the input graph G. Let there be ℓ such components that partition the nodes in V into parts $V_1, \ldots, V_\ell$. Let $N_l = |\{v \mid v \in V_l \text{ and } \mathbf{a}(v) = 1\}|$. Then, the optimal solution can be constructed by picking one (arbitrarily chosen) node from each of the connected components with the k highest N_l scores. Within these k components, all the nodes have an equal probability of being picked as effectors. This observation makes the DM model inappropriate for realistic settings.

7.5 Finding effectors on trees

Here we show that the k-EFFECTORS problem can be solved optimally in polynomial time when the graph $G = (V, E, p)$ is a tree. For clarity, we denote such a graph by $\mathcal{T} = (V, E, p)$.

7.5.1 The optimal DP algorithm

Our polynomial-time algorithm uses dynamic programming. The main idea is the following: given a subtree whose root has δ children, the optimal way of specifying at most k effectors from this subtree must follow one of two patterns: in the first pattern, we include the root of the subtree to the set of effectors, and then recurse on the children with budget $(k - 1)$. In the second, we do not include the root of the subtree, and instead recurse on the children with a budget k.

A naive way of implementing the recursion would result in partitioning the δ children into k (or $k-1$) parts and taking the minimum-cost partition. However, when $\delta >> 2$, computing the cost of all possible partitions is expensive. To circumvent this, we make a simple transformation that converts any tree to a binary tree.

We construct the new tree $\mathcal{T}_b$ from the original tree $\mathcal{T}$ as follows: we start from the root of $\mathcal{T}$, *root*$(\mathcal{T})$. Suppose that v is an internal node of $\mathcal{T}$ with children $v_1, \ldots, v_\delta$, with $\delta > 2$. We replace v with a binary tree of depth at most $\log \delta$ and leaves $v_1 \ldots, v_\delta$. Picking each one of the leaves $v_1, \ldots, v_\delta$ introduces a cost calculated the way we described above. Recall that we have a budget of k effectors. Every node v_i that corresponds to an actual node in the original tree $\mathcal{T}$ uses one unit of the budget, if picked as an effector. Further, the newly-created internal nodes in $\mathcal{T}_b$ that do not correspond to any actual nodes in $\mathcal{T}$ can never be picked as initiators. Directed edges are added between node v and these new internal nodes, as well as between the internal nodes themselves. The direction is always from the root to the leaves and the weight of these edges is set to 1. In this way, the directed edges that are associated with the newly added internal nodes in $\mathcal{T}_b$ do not influence the propagation from v to its children. This transformation is repeated recursively for each child $v_1, \ldots, v_\delta$. We denote the set of newly added (dump) nodes by D.

The following two observations are a direct consequence of the above process. Moreover, they guarantee that the newly-created binary tree causes bounded increase in the number of nodes and the depth of the original tree.

Observation 1. *The number of nodes in the binary tree $\mathcal{T}_b$ is at most twice the number of nodes of tree $\mathcal{T}$.*

Observation 2. *If Δ is the maximum out-degree of a node in tree $\mathcal{T}$, then the depth of the binary tree $\mathcal{T}_b$ is at most a factor of $\log \Delta$ larger than the depth of $\mathcal{T}$.*

Following the proofs appearing in similar constructions for different problems [139, 140, 141] we can also prove the following observation.

Observation 3. *the optimal solution to the k-*EFFECTORS *problem on $\mathcal{T}_b$ is the same as the optimal solution of the k-*EFFECTORS *problem on tree $\mathcal{T}$.*

Intuitively, this is because the newly-added nodes in $\mathcal{T}_b$ can neither be picked as effectors nor influence the information-propagation process. This is because all their outgoing edges have weight 1.

Given the above transformation, we can always assume that our influence tree is binary and we use $\mathcal{T}$ to refer to such binary tree. For a node v of the tree, we use $\textsc{Opt}(v, X, k)$ to denote the cost of the best solution in the subtree rooted at node v, using at most k effectors; X simply keeps the effectors in the current solution. Finally, for a node v we use $r(v)$ ($\ell(v)$) to refer to the right (the left) child of node v. Then, we evaluate the following dynamic-programming recursion on the nodes of the tree $\mathcal{T}$:

$$\begin{aligned}
\textsc{Opt}(v, S, k) = \min & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (7.3)\\
& \Big\{ \min_{k'=0}^{k} \big\{\textsc{Opt}(r(v), S, k') + \textsc{Opt}(\ell(v), S, k-k') + C(v, S)\big\},\\
& C(v, S \cup \{v\}) + \min_{k'=0}^{k-1} \big\{\textsc{Opt}(r(v), S \cup \{v\}, k') +\\
& +\textsc{Opt}(\ell(v), S \cup \{v\}, k-k'-1\big\}\Big\}.
\end{aligned}$$

The first term of the dynamic-programming recursion corresponds to not choosing v to be in S and the bottom term corresponds to choosing v to be in S. In order to guarantee that no newly-added node in the set D is picked as an effector, we set $C(v, S) = \infty$ for every $v \in D$ and any $S \subseteq V$. In addition, since the effectors are always selected from active nodes we also add a similar check to guarantee that no inactive nodes are picked. We call this dynamic-programming algorithm the DP algorithm.

The first term of the dynamic-programming recursion consists of $2k$ lookups on precomputed values in the table $\textsc{Opt}$ and it thus takes $\mathcal{O}(k)$ time. The bottom term, however, needs to go through all the nodes in the subtrees rooted at $\ell(v)$ and $r(v)$ and compute the additional cost incurred by the addition of node v as an effector. In the worst case, there are $\mathcal{O}(n)$ such terms and there are $O(k)$ evaluations that need to be done. Therefore, the computation of a single entry in table OPT requires $\mathcal{O}(nk)$ time. This is an overestimate of the actual time since, on average, the expected size (number of nodes) of a subtree in a binary tree is $\mathcal{O}(\log n)$. Therefore, the expected time required for the evaluation of a single entry is $\mathcal{O}(k \log n)$. Given that there are kn different entries, the worst-case time complexity of the DP algorithm is $\mathcal{O}(n^2k^2)$, while the expected running time is $\mathcal{O}(k^2 n \log n)$. In the above analysis we have assumed that given $C(v, S)$ we can compute $C(v, S \cup x)$ in constant time. In fact, this can be done by keeping at every node v the value of the product $\prod_{s \in S} \left(1 - \prod_{(y \to z) \in \text{path}(x,v)} p(y \to z)\right)$. The addition of a new node x in S would then simply require the update of this product and the use of Equation (7.1) for computing $\alpha(v, S \cup \{x\})$

Such bookkeeping comes with increasing space requirements: apart from storing the $n \times k$ values of table OPT, we also need to store k values of the product per node. Therefore, the total space required by DP is $O(2nk)$.

Although the DP algorithm is optimal, its running time and space requirements may make it inappropriate for very large datasets. Therefore, we also propose two alternatives: the Sort and the OutDegree algorithms. Both Sort and OutDegree have subquadratic running times and require much less memory than DP. Further, our experiments on real data show that both algorithms perform almost as well as the optimal. However, one can construct examples and datasets in which the performance of these algorithms degrades.

7.5.2 The Sort algorithm

For a given tree $\mathcal{T} = (V, E, p)$, the Sort algorithm evaluates, for every node $v \in V$, the cost incurred when v is the only effector in $\mathcal{T}$. That is, for every node v the cost $C(\{v\}) = \sum_{x \in V} |\alpha(x, \{v\}) - \mathbf{a}(x)|$ is computed. The set of k effectors is then formed by

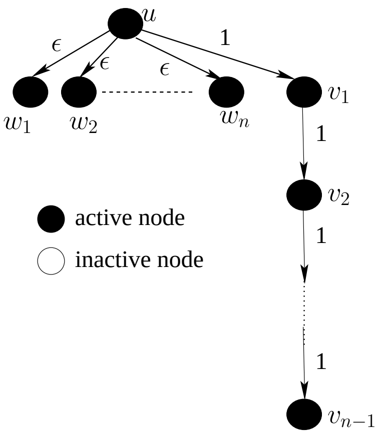


Figure 7.2: Influence tree with $2n$ active nodes. Edge weights are in $[0, 1]$ with $\epsilon \in (0, 1)$. The Sort algorithm for $k = n$ reports a solution that is $\mathcal{O}(n)$-times worse than the optimal.

picking the k active nodes with the smallest cost.

Computing the cost $C(\{v\})$ for every node $v \in V$ has worst-case running time $\mathcal{O}(n^2)$. However, the expected running time on binary trees is $\mathcal{O}(n \log n)$. Finally, the nodes are sorted based on their $C(\{v\})$ scores in $\mathcal{O}(n \log n)$ time.

Although Sort performs pretty well on real datasets, one can construct cases where the algorithm's performance is far from optimal. Consider for example the directed influence tree in Figure 7.5.2. The tree consists of $2n$ active (denoted by black) nodes. Nodes $w_1, \ldots, w_n$ are activated by the root u with probability ϵ. Root has influence probability 1 to v_1 and every node v_i activates node v_{i+1} also with probability 1 ($1 \leq i \leq (n-1)$). The cost of the root u is $C(\{u\}) = (1-\epsilon)n$. The cost of any node v_i (for $1 \leq i \leq n-1$) is $C(\{v_i\}) = i + n$. Similarly, the cost of activating one of the w_j nodes (for $1 \leq j \leq n$) is $C(\{w_j\}) = 2n - 1 > C(\{v_i\})$ for $1 \leq i \leq n-2$; for $i = n-1$ we have a tie in which case the algorithm resolves it by setting $C(\{v_{n-1}\}) < C(\{w_j\})$. Solving the k-EFFECTORS problem for $k = n$, the Sort algorithm would report as effectors $S = \{u, v_1, \ldots, v_{n-1}\}$, with cost $C(S) = n$. However, the optimal set is $S^* = \{u, w_1, \ldots, w_{n-1}\}$, with cost $C(S^*) = (1-\epsilon)$. Therefore, the performance ratio of Sort is $\frac{n}{(1-\epsilon)}$. This is a ratio of order $\mathcal{O}(n)$. Thus, Sort gives solutions that are at least $\mathcal{O}(n)$ times worse than the optimal.

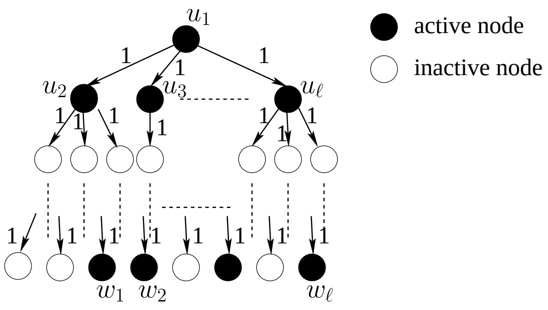


Figure 7.3: Influence tree with 2ℓ active and $n - \ell$ inactive nodes. All edge weights are equal to 1. The `OutDegree` algorithm for $k = \ell$ reports a solution that is $\mathcal{O}(n)$ times worse than the optimal.

7.5.3 The `OutDegree` algorithm

For tree $\mathcal{T} = (V, E, p)$, the `OutDegree` algorithm picks the k active nodes with the highest weighted out-degree in the influence tree $\mathcal{T}$. The complexity of the algorithm is defined by the computation of these degrees and the time required for sorting them. Therefore, the total running time is $\mathcal{O}(n + n \log n)$.

Our experiments with real data indicate that there are many cases where `OutDegree` performs well in practice. However, there are also cases, where the solutions reported by `OutDegree` are far from optimal. For example, consider the influence tree in Figure 7.5.3. The tree has n nodes, 2ℓ of which are active (black nodes) and $(n - 2\ell)$ are inactive (white nodes). That is, apart from nodes $u_1, \ldots, u_\ell$ and $w_1, \ldots, w_\ell$ all other nodes are inactive. For this tree, we also assume that all edges go from nodes closer to the root to nodes closer to the leaves of the tree and all weights are equal to 1. If we use `OutDegree` to solve the k-EFFECTORS problem with $k = \ell$, the algorithm will report solution $S = \{u_1, \ldots, u_\ell\}$, with cost $C(S) = (n - 2\ell)$. On the other hand, the optimal solution is $S^* = \{w_1 \ldots, w_\ell\}$, with cost $C(S^*) = \ell$. This is because none of the $(n-2\ell)$ inactive nodes will get activated. Therefore, `OutDegree` can report solutions that are $\frac{n-2k}{k} = \mathcal{O}(n)$ times worse than the optimal.

7.5.4 Finding effectors in forests

So far, we have assumed that the influence tree is connected. Here, we show how to allocate the budget of k effectors among the trees of an influence forest. We show that this can be achieved by another dynamic-programming recursion.

If we use $\mathcal{F}$ to denote this forest consisting of L trees, $\mathcal{T}_1, \ldots, \mathcal{T}_L$, then we need to find the optimal way of distributing the k effectors to these L trees. Recall that, if for a tree $\mathcal{T}_i$ we assign budget $k_i \leq k$ effectors, then we can compute the optimal set of k_i effectors on this tree using the dynamic-programming recursion given by Equation (7.3). Let $\textsc{Opt}(\mathcal{T}_i, k_i)$ be the solution obtained using the DP algorithm on tree $\mathcal{T}_i$. Then, the optimal solution on forest $\mathcal{F}$ is calculated again using dynamic programming: let $\mathcal{T}_1, \ldots, \mathcal{T}_L$ a random but fixed ordering of the trees in the forest $\mathcal{F}$ and $\textsc{Gl}(\ell, c)$ be the cost of the optimal assignment of $c \leq k$ effectors on the first ℓ trees $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$. Then, $\textsc{Gl}(L, k)$ will give the optimal solution to our problem. The values of the $\textsc{Gl}$ table are given using the following dynamic-programming recursion:

$$\textsc{Gl}(\ell, c) \quad = \quad \min_{0 \leq c' \leq c} \textsc{Gl}(\ell - 1, c - c') + \textsc{Opt}(\mathcal{T}_\ell, c').$$

This dynamic programming recursion is a generic method of allocating the budget of k effectors to the connected components of the input graph. We presented it here for the case of forests because for trees we can compute $\textsc{Opt}(\mathcal{T}_i, c)$. However, this computation cannot be done (or approximated) in polynomial time within each component of an arbitrary graph (see Lemma 2 and Lemma 3).

7.6 Extracting the influence tree

While the input influence graphs may not be trees, we show here how one can extract an influence tree from an arbitrary graph. Given $G = (V, E, p)$, our goal is to extract the influence tree $\mathcal{T}$ that captures most of the information in G. We quantify the optimization problem using a maximum-likelihood approach.

For a tree $\mathcal{T} = (V_T, E_T, p)$ with $E_T \subseteq E$, we compute the *likelihood* of $\mathcal{T}$ as follows:

$$L(\mathcal{T}) \;\; = \prod_{(u \to v) \in E_T} p(u \to v).$$

Therefore, our goal is to extract the influence tree $\mathcal{T}$ that maximizes $L(\mathcal{T})$. In fact, instead of maximizing the likelihood we minimize the negative log-likelihood. That is,

$$\textsc{mLL}(\mathcal{T}) \;\; = \;\; - \sum_{(u \to v) \in E_T} \log p(u \to v). \tag{7.4}$$

Our approach for constructing the influence tree is query-dependent. That is, given the set of active nodes in G, we extract the influence tree $\mathcal{T}$ that *spans all the active* nodes in G and minimizes Equation 7.4. We call this subproblem the ACTIVE TREE problem and the extracted influence tree the *active tree* of G.

Unfortunately, solving the ACTIVE TREE problem is NP-hard. In fact, the problem is identical to the DIRECTED STEINER TREE problem. In the DIRECTED STEINER TREE problem the input consists of a directed weighted graph $G' = (V', E')$ a specified root $r \in V'$ and a set of terminals $X' \subseteq V'$. The objective is to find the minimum-cost tree rooted at r and spanning all the vertices in X' (i.e., r should have a path to every vertex in X'). Our setting is identical; the required nodes are the set of active nodes in G.

Here, we use the following efficient heuristic for constructing the *active* tree for a given influence graph: first, we construct the set of nodes R that consist of all the nodes in V that have no incoming edges. For each such root node $r \in R$ and for each node $s \in S$ we compute the shortest path from r to s in $\mathcal{T}$. Let $\mathcal{T}(r)$ be the tree consisting of the union of the edges in such shortest paths for the root node r. We then report as a solution the tree $\mathcal{T} = \operatorname{argmin}_{r \in R} w'\left(\mathcal{T}(r)\right)$. We call this simple algorithm the `dSteiner` algorithm.

So far we have assumed that the influence graph G is connected. If this is not the case we can follow one of the following two alternatives: (a) Find the strongly-connected components of the graph and then apply `dSteiner` independently in every component. This would output a forest of influence trees and, therefore, we can use the method described in Section 7.5.4. (b) Introduce an artificial node to the input graph is connect it via very low probability edges to all the nodes. This guarantees connectivity and allows us to use the `dSteiner` algorithm directly on this enhanced graph.

`dSteiner` can be replaced by any other approximation algorithm proposed for the directed Steiner tree problem [142, 143]. However, since the focus of our work is not on the study of methods for the directed Steiner tree problem, we only use the `dSteiner` algorithm for our experimental evaluation. `dSteiner` requires a simple all-pairs shortest path computation and it is much less computationally demanding than the majority of other existing methods for the same task.

Discussion: Our approach for extracting the influence tree from the influence graph finds the most probable tree that spans all active nodes. Therefore, different activation vectors lead to different trees. We believe that for large graphs, where only some of the nodes are active, it makes sense to extract influence trees that are item-dependent. Alternatively, one could construct the influence tree to be item-independent. That is, one could try to extract from G the tree that spans *all* the nodes in G and minimizes Equation (7.4). This problem is equivalent to solving the directed minimum-cost spanning tree (D-MST) problem on a directed graph G. Such a tree can be extracted using a polynomial-time solution to the D-MST problem [144, 145]. It can then be used for all activation vectors. The performance of this approach depends on the portion of active nodes in the input activation vector. For a small number of active nodes, it would create influence forests with a small number of active nodes per component. In fact, further experimental analysis verified this intuition. Due to space constraints we do not report these results here.

7.7 Experiments

In this section we evaluate the proposed algorithms for the k-EFFECTORS problem using the co-authorship graph extracted from the DBLP data. Our evaluation focuses on (a) showing indicative results from our methods and (b) evaluating the quality of the results with respect to the objective function.

7.7.1 The DBLP dataset

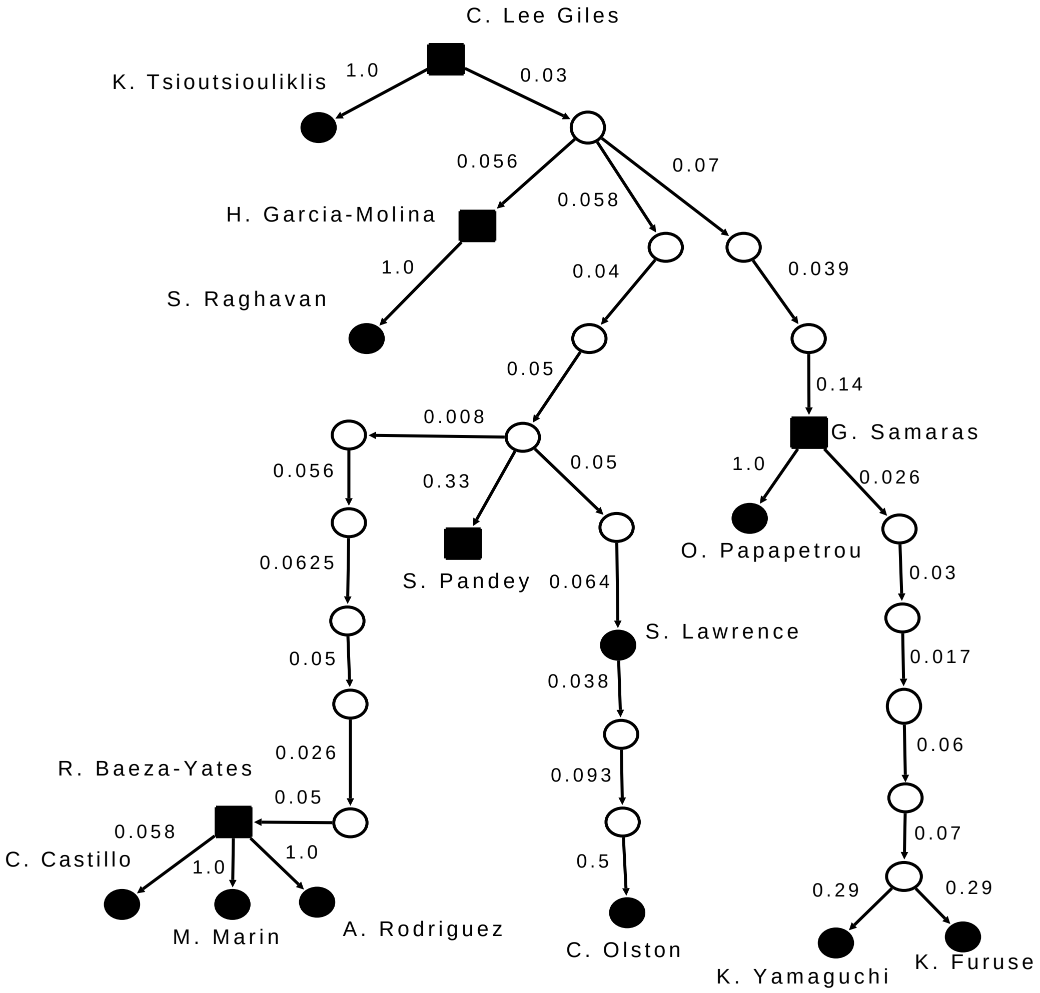
Using a snapshot of the DBLP data taken on April 12, 2006 we create a benchmark dataset for our experiments. We only keep entries of the snapshot that correspond to papers published in the areas of *Database* (DB), *Data mining* (DM), *Artificial intelligence* (AI) and *Theory* (T) conferences. Given this snapshot we create the network $G_{\text{dblp}} = (V, E, p)$ as follows: nodes in V correspond to authors; an author is included in V if she has at least three papers in the data. Each author i is associated with a set of terms S_i; these are the terms that appear in at least two titles of papers that i has co-authored. This process creates $|V| = 5508$ individuals and a set of 1792 distinct terms. Each term $t \in S_i$ is also associated with a timestamp $T_i(t)$, i.e., the year first used by author i. Two authors i, i' are connected by an edge in G_{dblp} if they co-authored at least two papers. The weight of the directed edge $(i \rightarrow i')$ is computed using the following simple rule:

$$p(i \rightarrow i') \quad = \quad \frac{|\{t \mid t \in S_i \wedge t \in S_{i'} \wedge T_i(t) < T_{i'}(t)\}|}{|S_{i'}|}$$

That is, we compute the probability that an item appearing in i' is a result of the influence of node i on i'.

We focus our experiments on activation vectors for 15 terms that correspond to research themes in computer science. The list of these 15 terms is shown in the first column of Table 7.1. For each term q, we extract the corresponding activation $\mathbf{a}_q$ so that $\mathbf{a}_q(i) = 1$ if

Figure 7.4: Influence Tree $\mathcal{T}_q$ for $q =$"crawling". The tree is extracted from the original G_{dblp} influence graph.



$q \in S_i$. Given graph G_{dblp} and activation vector $\mathbf{a}_q$, we compute the active tree associated with q, denoted by $\mathcal{T}_q$, using the `dSteiner` algorithm (see Section 7.6). We always use those trees to identify the set of effectors using one of the three effector-finding algorithms for trees: `DP`, `OutDegree` and `Sort`.

7.7.2 An illustrative example

We start by showing an indicative output of our approach on the tree $\mathcal{T}_q$ for $q =$"crawling".[3] The active tree $\mathcal{T}_q$ is shown in Figure 7.4. The black nodes are active with respect to the term, while the white nodes are inactive. Next to every active node, we show the name of the author it represents. For every edge we also display its weight in $\mathcal{T}_q$. We extract the effectors on this tree using DP and $k = 5$. The black square nodes are the 5 effectors chosen by the algorithm. The choices made by the algorithm are intuitive. C. Lee Giles covers K. Tsioutsioukliklis whom he influences with high probability. Similarly, H. Garcia-Molina covers S. Raghavan and G. Samaras covers O. Papapetrou. R. Baeza-Yates is also picked as an effector, due to the high influence probabilities to his co-authors. S. Pandey is the only leaf node chosen as an effector; this is simply because there are no high-probability paths to him from other active nodes. On the other hand, there are some active nodes (e.g. C. Olston) that are not chosen as effectors. This is because there was not enough budget and the algorithm determined that selecting the other nodes benefited the objective function. In fact, when we increased the budget to $k = 6$, C. Olston was the only new addition to the set of effectors. For $k = 7$, K. Furuse was also included, even though choosing K. Yamaguchi would clearly result in the same overall cost.

7.7.3 Comparison of effector-finding algorithms

This section evaluates the different algorithms for the k-EFFECTORS problem with respect to the cost function $C()$. We use the activation vectors for the 15 terms shown Table 7.1 and construct the 15 different active trees (one tree per term). Then, we run the DP, Sort and OutDegree algorithms on each of the 15 trees. In addition to these three algorithms, we also evaluate Random; an algorithm that randomly picks the effectors on a given input tree. The performance of all the algorithms with respect to the objective function and $k = 10$, is

[3] The choice of the term was guided by the size of its active tree, which proved small enough to visualize.

Table 7.1: Cost of the solutions reported by DP, OutDegree, Sort and Random algorithms on the active trees for 15 distinct terms.

Term	DP	OutDegree	Sort	Random
collaborative filtering	31.40	34.19	34.19	40.13
graphs	558.75	560.41	558.75	582.92
wavelets	16.39	16.73	17.40	19.95
pagerank	2.33	4.20	4.20	4.20
privacy	47.09	47.56	50.22	59.59
clustering	514.94	520.74	519.10	560.99
classification	343.54	344.44	343.54	361.86
xml	382.59	385.29	382.59	418.01
svm	20.29	21.15	21.15	27.92
crawling	0.49	3.07	4.03	4.07
semisupervised	25.25	25.45	25.31	30.66
boosting	86.02	89.08	86.02	98.82
microarrays	24.35	28.93	29.07	42.46
streams	275.72	279.16	279.68	300.84
active learning	11.62	12.49	12.49	18.55

shown in Table 7.1. Recall that, since our problem is one of cost minimization, the lower the value, the better the performance of the algorithm. Also, since DP is optimal, its cost serves as the baseline for the other algorithms.

As we can see from the table, the Random algorithm is clearly worse than the others for all 15 terms. In contrast, the Sort and OutDegree algorithms report solutions with costs consistently close to the optimal (achieved by the DP algorithm), for most of the terms in the table.

The near-optimal performance of Sort and OutDegree is clearly beyond the expectations set by the worst-case analysis presented in Sections 7.5.2 and 7.5.3. This can be explained by the structure of the G_{dblp} graph: many prolific and highly influential authors are also good effectors, particularly on terms with a large number of active nodes. This clearly helps Sort and OutDegree, since they favor such nodes.

In order to further explore the behavior of the same algorithms under different scenarios we proceed as follows: first, we generate the active tree $\mathcal{T}_q$ for each of the terms in Table 7.1,

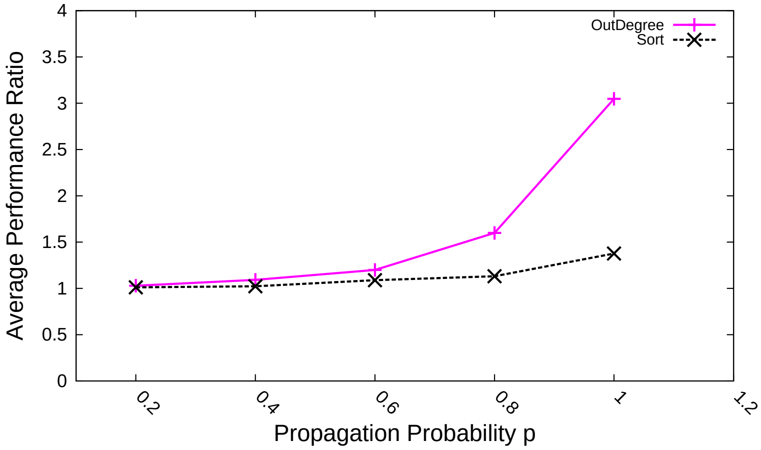


Figure 7.5: Average performance ratio of `Sort` and `OutDegree` for 15 trees $\mathcal{T}_q$ with modified influence probabilities p set uniformly across all the edges; $p \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$

as we did for the previous experiment. Then, we replace the actual influence probabilities with some constant probability $p \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$. Therefore, for every term q, we construct 5 different instances of the $\mathcal{T}_q$ tree and apply the `DP`, `Sort` and `OutDegree` algorithms on each of them. Our motivation is to moderate the effects of highly influential nodes, thus making it more challenging for the algorithms to identify the set of effectors. In Figure 7.5, we plot the average *performance ratio* of the algorithms (ratio of the cost of the solution reported by an algorithm divided by the cost of the optimal solution reported by `DP`), for the different values of p. The average is taken over all the 15 different trees. Naturally, the closest the ratio is to 1, the closer the solution is to the optimal.

The results show that the performance of `OutDegree` and `Sort` deteriorates as the value of p approaches 1. In particular, for higher values of p, the performance ratio of `OutDegree` is significantly higher than 1. This can be explained by the fact that, as the edge weights approach 1.0, an increasing number of nodes gain high-probability paths to many other nodes. As a result, the weighted criterion used by `OutDegree` to pick effectors loses its advantage and the performance of the algorithm deteriorates. A similar argument

Table 7.2: The $k = 10$ effectors reported by the DP algorithm for terms $\{$graphs, XML, Collaborative Filtering$\}$.

Graphs	XML	Collaborative Filtering
A. Brandstadt	A.Zhou	A. Nakamura
A. Z. Broder	D. Srivastava	B. Mobasher
C. Faloutsos	E. A. Rundensteiner	D. Heckerman
D. Peleg	F. Bry	D. Poole
F. Hurtado	H. V. Jagadish	F. Yang
F. T. Leighton	J. Srinivasan	H.-P. Kriegel
N. Linial	M. Krishnaprasad	J. M. Kleinberg
N. Alon	O. Diaz	M. Li
S. Leonardi	S. Pal	R. S. Zemel
W. Wang	T. Milo	W. Du

can be made for `Sort`: the algorithm favors nodes with high-probability paths to the active parts of the network. When such paths exist for most of the nodes, it becomes harder for the algorithm to identify the optimal set. The variance values of the ratios reported in Figure 7.5 for $p = \{0.2, 0.4, 0.6, 0.8, 1\}$ are $\{0.02, 0.04, 0.18, 0.23, 0.51\}$ for `Sort` and $\{0.03, 0.14, 0.28, 0.89, 2.55\}$ for `OutDegree`, respectively. Note that `OutDegree` is more susceptible than `Sort` in making incorrect choices as the value of of p increases. As a result, we observe larger values of variance in the ratios observed by `OutDegree`.

7.7.4 Qualitative evidence

Next, we present qualitative evidence of the results obtained by optimally solving the k-EFFECTORS problem on G_{dblp}. Our motivation is to show that, in a realistic setting, the results we obtain are reasonable and intuitive. Table 7.2 shows the results obtained using the optimal DP algorithm to solve the k-EFFECTORS problem on three different $\mathcal{T}_q$ trees for $k = 10$. We report the results for three terms $q = \{$graphs, XML, and Collaborative Filtering$\}$. We purposefully select terms from three popular -albeit different- areas of computer science, in order to capture results coming from diverse parts of the G_{dblp} graph.

A quick observation indicates that the reported sets of effectors include some very prolific authors. This can be verified by checking the overall number of papers per author, as recorded in the **DBLP** dataset. In fact some of the authors have over 150 papers (e.g,. D. Peleg (213), N. Alon (250), H. V. Jagadish (156), E. A. Rundensteiner (191), H.-P. Kriegel (214)). The number of papers serves as an indicator of the author's influence in the graph. Authors with more papers typically have more distinct coauthors and are active with respect to more terms. Also, recall that we operate on the active tree $\mathcal{T}_q$, extracted so that it mostly consists of active nodes associated with a term. As a result, prolific authors are likely to be chosen as effectors, since they have high-probability paths to many of these active nodes. However, authors with relatively small number of papers are also included as effectors. An intuitive explanation for this is the following: even though well-connected nodes can be reasonable effectors that explain a large part of the observed activation vector, they are also more likely to be connected to inactive nodes. As a result, selecting only highly-connected nodes as effectors increases the overall cost of the solution. Overall, the set of effectors can include nodes of variable connectivity and influence, as long as they can best describe the given activation state of the network.

Although the reported effectors per term are all from the general area of computer science indicated by the term itself, each one of them covers a different sub-community. For example, for the term "Collaborative Filtering", we can see D. Heckerman and D. Poole – both effectors for the machine-learning community – J. Kleinberg – an effector for the theory community – and H-P. Kriegel – an effector for the database community. Further, many of the effectors come from different geographical regions, and, thus, act as effectors for different sets of authors. In fact, further analysis showed that the reported effectors have small overlap in their sets of co-authors.

Similar observations can be made for the other two terms. For example, for the term "graphs", C. Faloutsos is an effector for the data-mining community, while the majority of the other authors are effectors that cover different parts of the theory community. Again, the number of common co-authors between every pair of the reported effectors is very small.

7.8 Conclusion

Given a network where a subset of the nodes are active, and a probabilistic propagation model, we defined the problem of finding the subset of active nodes that best explain the observed activation state. We called these nodes *effectors*. We studied the complexity of the k-EFFECTORS problem in directed graphs and trees. For general directed graphs, we showed that the k-EFFECTORS problem is NP-hard to solve or even to approximate. However, we showed that for directed trees the problem can be solved optimally in polynomial time via dynamic programming. We also presented a general framework, where, given a directed influence graph and an activation vector, we first extract the most probable active tree that spans all the active nodes in the network. We then use the dynamic-programming algorithm to identify the optimal set of effectors in this tree. In our experimental evaluation, we demonstrated that our algorithms perform well with respect to our objective function. The reported sets of effectors provide useful insight about the network and the interactions between the nodes. In the future, we plan to further explore the utility of effectors in other types of networks, including computer and biological graphs.

Chapter 8

Interactive Recommendations in Social Endorsement Networks

8.1 Introduction

The desire of users to exchange information and share their personal opinions has been one of the main causes of the astounding popularity of social networks. Users use social networks to comment on a variety of different *entities*, such as photos, movies, products, or even other users. In many popular platforms this method of expression has been formalized, allowing users to express their approval of an entity by *endorsing* it. On `Facebook.com`, users have the option to "like" a photo, video or text message that has been posted by another user. In the same platform, users can become "fans" of an entity by simply joining the respective fan-group. The nature of such fan-groups is impressively diverse, including groups for real-life celebrities, commercial products, popular TV shows or even campaigns (e.g. a group promoting cancer awareness). Another relevant example is `CiteULike.org`, where users can show their approval of a published paper by including it in their "Library". Further, on `Twitter.com`, users can express their interest and approval by becoming "followers" of other users.

By visualizing an endorsement as an edge from a user to an entity, we can view a *Social Endorsement Network* as a bipartite graph $G = (U, V, E)$, where U is the set of users, V is the set of entities, and E is the set of endorsement edges. A problem that naturally arises in such a network is recommending to the user other entities that he is likely to be interested in. As we show in our work, the information encoded in the graph of the social endorsement network can serve as an exceptional foundation for a solution to this problem. The intuition is simple: an endorsement serves as a verification that the user approves the endorsed entity. Examining the set of entities that are endorsed by a single user can provide some information on his preferences, but it does not answer the most important question: *Why did the user choose to endorse this particular entity?* To solve this question, we call upon the wisdom of crowds: first, we find groups of entities that are endorsed by the same large groups of users. For each group, we then examine the common characteristics of the included entities and identify the aspects that truly appealed to the same large set of users. The product of this first phase is a collection of groups, where for each group we have a set of *tags* that encode its most attractive and characteristic aspects. Given this information, the next step toward a great recommendation framework comes naturally: we make our system *interactive*, allowing the user to specify his own personal interests in the form of a query. The submitted query is then streamed through the mined groups, in order to identify those that best match the user's interests. The main problem addressed in this work is the following:

Problem 10. Given a user-submitted query and a social endorsement network G, we want to identify and recommend groups of entities that match the query and also share a significant number of common endorsers.

In order to accurately encode the user's preferences, we formalize queries as sets of tags (keywords). This is an intuitive and flexible method, with which practically every user is familiar. Below are some examples of relevant queries, as could be formed in popular social networking platforms:

- `Twitter`: Recommend *female* *singers* from *the USA*.
- `Facebook:` Recommend (fan pages of) *Chinese Restaurants* *in San Fransisco*
- `CiteULike:` Recommend research papers on *Social Networks*

The underlined terms represent the query-tags that encode the user's interests. A possible recommendation of our framework for the 1st query is : "Whitney Houston, Mariah Carey and Celine Dion match the query and also share 50,000 followers". The interaction with the user and the authority offered by the large number of common endorsers make our recommendations *explainable* and intuitive to the user. Explainable recommendations are increasingly popular and have been the focus of numerous research efforts [146].

A diagram of our framework is shown in Figure (8.1): Given a social endorsement network, we first extract groups of entities that share a significant number of common endorsers. Next, we identify the appropriate set of tags for each of the reported groups. Taking the assigned tags into consideration, we then apply a filter that eliminates redundant groups (i.e. groups that can be induced by others), and produces a compact and informative corpus. The final corpus is then organized in an appropriate index structure, which, together with an efficient algorithm for query evaluation, compose a search engine able to recommend appropriate groups any for multi-tag query submitted by a user.

Contribution: Our work is the first to formalize and solve the problem of *interactive recommendations in social endorsement networks*. We thoroughly discuss the architecture of the proposed framework and demonstrate its efficacy through a thorough experimental evaluation on real datasets. The benefits of our framework are clear:

- It is interactive, allowing the user to repeatedly query the system for different types of entities.
- Its principled and efficient architecture make it ideal for large-scale systems.

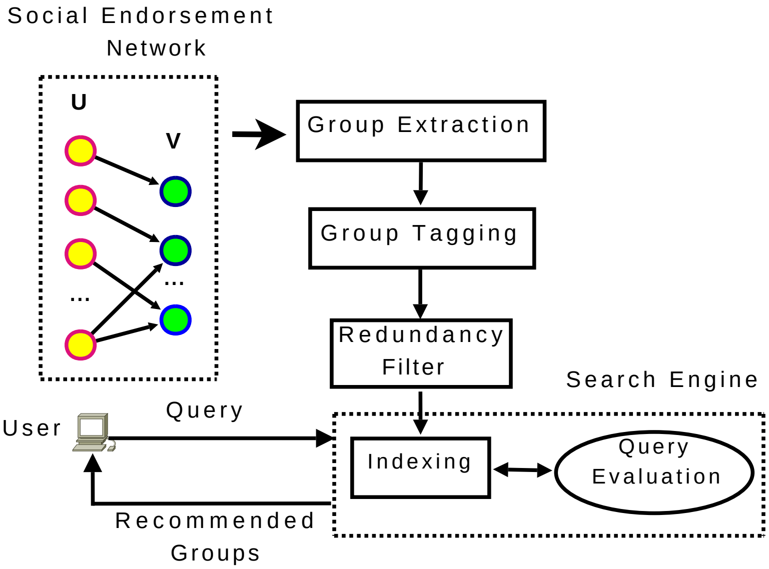


Figure 8.1: A diagram of our complete search framework

- The recommended entities come organized into groups, with each group representing a different cohesive set of similar entities.
- The recommendations are easily *explainable* and, thus, more intuitive to the user.

Another significant contribution of this work is the release of a brand new dataset (crawled from `Twitter.com`), which is ideal for research on social endorsement networks.

8.1.1 RoadMap

We begin in Section 8.2 with an overview of the related work. In Section 8.3, we discuss the identification of popular groups of entities. Then, in Section 8.4, we discuss the process of tagging the reported groups. In Section 8.5 we introduce a principled filtering method for the elimination of redundant groups. In Section 8.6 we describe the interactive recommendation mechanism, consisting of an appropriate index structure and an efficient query evaluation algorithm. In Section 8.7, we illustrate the efficacy of our methods through a thorough experimental evaluation on real datasets. Finally, we conclude in Section 8.8, with a brief overview of our work.

8.2 Related Work

To the best of our knowledge, ours is the first work to consider the problem of *interactive recommendations in social endorsement networks.* Nonetheless, our work has ties to numerous fields. Next, we present a brief overview of the relevant literature.

Our recommendation system is defined in the context of a social endorsement network, which we formalize in the work. Social endorsement has been also considered in the past, albeit in a different context. Kunegis et al. [147] analyze different aspects of the social graph from `Slashdot.org`, where users have the option to tag others as "friends" or "foes", thus providing positive or negative endorsements. In another relevan paper Leskovec et al. [148] discuss the prediction of positive and negative edges in social networks.

Different types of recommendation systems have been proposed in the broad context of social networks: Guy et al, considered *the familiarity network* among the different users to support their recommendation system [149]. In a related paper, Bonhard et al. [150] explore how the familiarity and similarity among users can be utilized to improve recommendations.

The first phase of our framework utilizes a module for mining frequent (popular) groups of entities. Pattern mining has been explored in the context of recommendation systems [151, 152, 153], albeit in contexts that are completely different to our social network paradigm. In the second phase of our system, we employ a type of *social tagging*. Tagging is an increasingly popular research topic, mainly due to the success of social networking platforms that give their users the option to tag different objects (e.g. photos, videos). A significant amount of work has been devoted to methods for automatic tag extraction [154, 155, 156, 157] and to using tagging to enhance recommendation systems [158, 159, 160, 161, 162, 163].

Users can interact with our recommendation system via queries. Interactivity in the context of recommendations systems has also been studied in the past: Viappiani et al. [164] propose a conversational recommender that collects information and adapts to the user's preferences. In a similar setup, Bridge at al. [165] try to identify the most suitable items for a user, while keeping query updates to a minimum. Schenkel et al. [166] propose an incremental top-k querying-algorithm that takes into consideration the relationships among users to rank tagged objects (e.g. photos).

Finally, our work has ties with collaborative filtering, an extensively studied problem in the context of recommendation systems [167, 168, 169, 170]. Though relevant, our work is the first to focus on social endorsement networks and enables *interactive* and *explainable* recommendations.

8.3 Extraction of Popular groups

In this section, we describe the process of identifying groups of entities with a significant number of common endorsers, given a social endorsement network. This is only the first phase of our framework, albeit an important one, since it produces an initial collection of popular entity-groups. These groups will then be processed, tagged, filtered, and finally organized toward an efficient recommendation engine.

We formalize the problem of extracting popular groups as an instance of the problem of mining frequent itemsets: we are given a set of transactions, where each transaction includes a set of items. We then want to find groups of items that were often grouped together. In our context, a transaction is the set of entities that are endorsed by a user. Formally, we define the problem as follows:

Problem 11. *Given a social endorsement network $G = (U, V, E)$ and a group of endorsed entities $g \in 2^V$, let $N(g)$ return the set of common endorsers of g in G. Then, find the set*

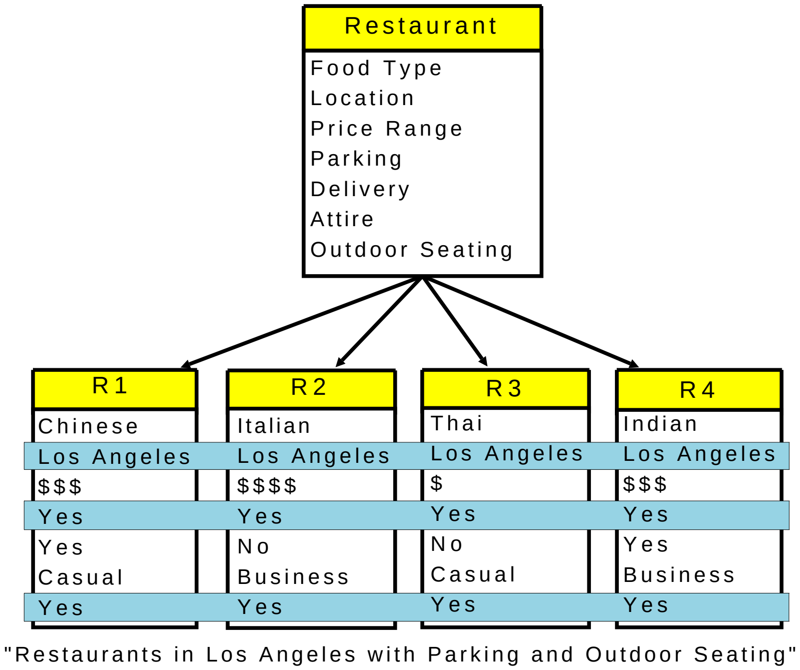


Figure 8.2: An example of using tags to identify the correlation among the entities in a group. In this case, the four entities are all Restaurants in Los Angeles that offer both Parking and Outdoor seating.

of entity-groups $\mathcal{G}$, so that

$$\mathcal{G} = \{g \mid g \in 2^V,\ |g| \geq 2,\ |N(g)| \geq T\} \tag{8.1}$$

As formulated above, the problem asks for all groups of at least two endorsed entities that have at least T endorsers in common. In the Experiments section, we show how tuning the value of T affects the number of reported groups.

By representing the set of entities endorsed by each user as a transaction, Problem 11 can be efficiently solved by any of the popular algorithms for mining frequent itemsets. In our experiments, we use the algorithm proposed in [171], which proved efficient enough to easily handle a database of over six million transactions.

8.4 Group Tagging

After we obtain the popular groups, the next step is to tag them in a way that facilitates search. Given a group, we want to answer the following question: *Why where these entities endorsed by the same large set of users?* To answer this, we need to identify the common characteristics that make these entities appealing to the same crowd. In order to achieve this, we need to obtain and record information on the different attributes of each entity. For example, if the endorsed entity is a restaurant, the list of attributes may include the type of food served or the restaurant's location. Such information can be easily encoded in the form of *tags*. Tagging is an increasingly popular feature, available in many social networking platforms. For example, in Flickr and Facebook, users can tag photos and videos with descriptive terms or phrases of their choice. Such tags can be submitted by users who manually assign descriptive tokens to each entity, or produced by an automated tagging method [156, 155, 154, 157]. **Our framework is compatible with any tagging method** that can assign a set of tags $ts(e)$ to each entity e. These TagSets can be then used to compute the TagSet $ts(g)$ of an entire group $g = \{e_1, e_2, ...\}$ as follows:

$$ts(g) = \bigcap_{e \in g} ts(e) \tag{8.2}$$

Even though this definition worked superbly in our experiments, it can easily be relaxed to include tags that appear in **a large majority of the group's entities, rather than all of them**.

Getting the Tags:In the case of automated tag-extraction, a question that arises is the following: *Where can we mine the required TagSets from?* Typically, automated methods are based on a piece of descriptive textual information that is available for each entity. In cases where the entity is itself consisting of text (e.g. a webpage or other document), then obtaining such information is a non-issue. Given the abundance of information that are available on the Web, such text summaries can be easily obtained for virtually any type of entity: Facebook Groups and Fan Pages have a short passage describing the nature and purpose of the group. On Twitter and MySpace, users provide a self-written description in their profiles. Informative pieces of text can also be extracted from sources outside the network: if the endorsed entity is a product, the text from the product's official website can serve as a descriptive summary. If the entity is a movie, the source can be the plot summary from sites like `imdb.com`. If the entity is an influential person, we can use the text from his personal page or the respective entry on sites like `Wikipedia.com`.

Structured Content: In many cases, informative content can be found in a semi-structured format in the Web. The templated entries on the right side of the pages on Wikipedia.com serve as a characteristic example of such a format. Such templates facilitate the direct extraction of informative tags. An intuitive way to compose TagSets from such data is via the construction of *profiles*. Examples are given in Figures (8.2) (*Restaurant*) and (8.3) (*Athlete, Singer, Politician*). The profiling process adds an additional level of abstraction and facilitates the grouping of different entities; Since the available entities are evaluated on a fixed set of attributes, it is easier to identify common characteristics and decode the correlation among the members of a group. An illustrative example is given in Figure (8.2): we are given a profile representing a restaurant, along with a group of four matching entities. In this case, since all the entities in the group belong to the same profile, the attribute-set of the group includes only the seven attributes of the profile: {*Food Type, Location, Price Range, Parking, Delivery, Attire, Outdoor Seating*}. Then, the TagSet of the group will contain the attribute values that remain the same for all restaurants. In this case: $ts(g) =${*Los Angeles, With Parking, With Outdoor Seating*}. These three tags compose the TagSet of the group , and reveal why such a large number of users endorsed all 4 restaurants.

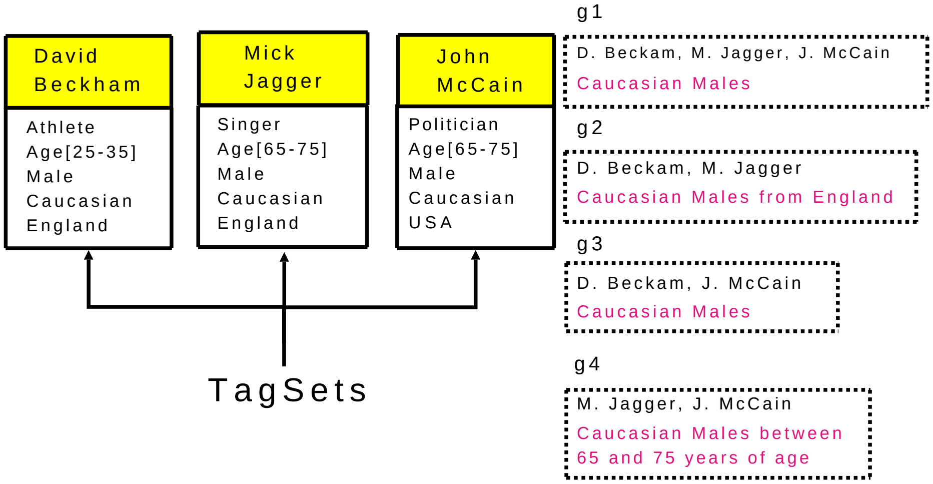


Figure 8.3: An Example of Group redundancy: g_3 is pruned, since it is a subset of g_1 and $ts(g_3) \subseteq ts(g_1)$.

8.5 Eliminating Redundancy

In this section we identify a type of redundancy among entity-groups and propose a principle method to eliminate it. Consider the example given in Figure (8.3): we are given three entities: David Beckham (athlete), John McCain (politician) and Mick Jagger (singer). We assume that the three individuals have a significant number of common endorsers and have been identified as a popular group. On the right, the figure shows all the possible (sub)groups with at least two members, along with their respective TagSets. We observe that g_3 is a subset of g_1, and also bears no additional tags (i.e. $g_3 \subset g_1$ and $ts(g_3) = ts(g_1) = \{Caucasian, Male\}$). Therefore, g_3 is redundant and we can safely prune it without losing any information. On the other hand, even though both g_2 and g_4 are also subsets of g_1, they also have richer TagSets and thus have to be included in the final set. Formally, we define the problem as follows:

Problem 12. *Given a set of groups $\mathcal{G}$, find a filtered set $\mathcal{G}^* \subseteq \mathcal{G}$, so that:*

1. $\forall g \in \mathcal{G},\ \exists g' \in \mathcal{G}^* s.t.\ \{ts(g) \subseteq ts(g')\ and\ g \subseteq g'\}$

2. *$\mathcal{G}^*$ is the smallest set among all those that satisfy the 1st condition.*

Algorithm 6 `GroupFilter`

```
Input: Set of Entity Groups $\mathcal{G}$
Output: Filtered set of non-redundant Groups $\mathcal{G}^*$
$filteredIndex \leftarrow \emptyset$ // supports superset queries.
Sort $\mathcal{G}$ in desc order by group size
for each group $g \in \mathcal{G}$ do
    $isRedundant \leftarrow false$
    $\mathcal{S} \leftarrow lookup\_sups(filteredIndex, g)$
    for (each super-group $S \in \mathcal{S}$) do
        if $(ts(g) \subseteq ts(S))$ then
            $isRedundant \leftarrow true$
            $break$
    if $(!isRedundant)$ then
        $filteredIndex.insert(g)$
return $filteredIndex.getGroups()$
```

The first condition requires that, for every group $g \in \mathcal{G}$, there exists a group $g' \in \mathcal{G}^*$ that contains all the entities of g, and is also tagged with all the tags included in $ts(g)$ (among others). The second condition implicitly asks for a set consisting exclusively of non-redundant groups: even if a single redundant group exists in $\mathcal{G}^*$, we can safely prune it and thus get a set of smaller size. In order to address this problem, we propose the `GroupFilter` algorithm, which reports a filtered set, consisting only of non-redundant groups. The pseudocode is given in Algorithm (6).

Details of Algorithm (6): The input consists of the complete set of groups $\mathcal{G}$, while the output is a filtered set $\mathcal{G}^*$ of all non-redundant groups. The algorithm maintains an index of the non-redundant groups ($filteredIndex$). For every group g, we probe the index to retrieve the set of (non-redundant) super-groups (i.e. groups that contain, among others, all the entities included in g). Any structure that supports such *superset queries* can be used to build the index. We use the UBTree [172], a simple and efficient structure for indexing sets. We refer the reader to the original paper for more details on the structure.

`GroupFilter` begins by sorting all the groups by size (i.e. number of members), in descending order. This ensures that all super-groups of a redundant group will be evaluated before it is. Then, for each group $g \in \mathcal{G}$, we probe the index to retrieve its set of super-groups $\mathcal{S}$. If there exists a super-group $S \in \mathcal{S}$ that has all the tags of g (i.e. $ts(g) \subseteq ts(S)$), then g is redundant and can be ignored. Note that, since the TagSet of a group is guaranteed to contain all the tags included in any of its super-groups, it is sufficient to check if $|ts(g)| \leq |ts(S)|$. If there exists no superset S that satisfies this inequality, g is non-redundant and can be safely inserted in the index. At this point we know that g is non-redundant, otherwise it would have been pruned earlier (since the groups in $\mathcal{G}$ are sorted). This guarantees that our index only contains non-redundant groups, leading to a structure that is smaller and faster to probe. After all the groups have been evaluated, the algorithm returns the filtered set of non-redundant groups.

8.6 Interactive Recommendations

In this section, we describe a search engine for the recommendation of entity-groups. Conceptually, we want to respond to queries of the type: *"Find large groups of entities that share a set of tags $\{t_1, t_2, ..., t_m\}$, and also have a significant number of common endorsers'*. By asking for larger groups, we maximize the amount of information returned to the user, who can then further investigate the numerous entities in a group. Maximizing the number of endorsers would not be reasonable in our context, since it would lead to trivial, single-entity groups. We formalize the problem as one of top-k evaluation, as follows:

Problem 13. *Given a set of entity-groups $\mathcal{G} = \{g_1, g_2, ..., g_n\}$ and a query of tags $q = \{t_1, t_2, ..., t_m\}$, find the k largest groups from $\mathcal{G}$ that satisfy the following condition:*

$$ts(g) \cap t_i \neq 0, \; \forall t_i \in q \tag{8.3}$$

Conceptually, Problem 13 asks for the k largest groups that contain all the tags of the query in their respective TagSets. To address the problem, we use an inverted index

Algorithm 7 `TopKFinder`

```
   Input: Inverted Index $index$, query of tags $q = \{t_1, t_2, ..., t_m\}$, int $k$
   Output: set of $top - k$ matching groups
1: $TopK \leftarrow \emptyset$ // sorted, holds at most k elements
2: $\mathcal{L} \leftarrow \{Li \mid t_i \in q\}$
3: while $TopK.size() < k)$ do
4:     for ($every\ List\ L \in \mathcal{L}$) do
5:         $g \leftarrow getNext(L)$
6:         if $(g = \texttt{null})$ then $return\ TopK$
7:         else if $(L'[g] \neq \emptyset, \forall L' \in \mathcal{L})$ then
8:             $TopK.insert(g)$
9: return $TopK$
```

structure, mapping each tag to the list of groups that contain it. The groups in each list are primarily sorted in descending order by their size. In addition, a secondary sort is done by the number of endorsers, also in descending order. This ensures that, among groups of equal cardinality, those with the highest number of endorsers will have priority. Given the inverted index, we can retrieve the top-k results using a simple evaluation algorithm, shown in Algorithm (7).

The algorithm, which we refer to as `TopKFinder`, begins by retrieving the set $\mathcal{L}$ of group-lists that correspond to the m tags of the query. Then, for each list $L \in \mathcal{L}$, the $getNext(L)$ function is used to retrieve the next group under sorted access. The function returns `null` if L has been exhausted. For each candidate group g in L, the algorithm checks if it is also included in all other lists in $\mathcal{L}$. Each list is checked using a random access probe, supported by an appropriate structure. An example of such a structure is a hash-set, where each group is hashed by a label consisting of its tags (or tag IDs) in lexicographical order. If g is indeed included in all the lists, then it is included in the $top - k$. The algorithm continues, until k groups have been identified or until at least one of the lists has been exhausted (Line 6).

`TopKFinder` is essentially a simplified version of the popular Threshold Algorithm (`TA)` [32]. In the typical use case of `TA`, the score of each object (group) is different in every list. Therefore, the algorithm has to retrieve the respective scores of the object from all the lists, and compute the cumulative value. A score-based threshold mechanism is used as a termination criterion. In our case, this mechanism is redundant, since the score of each group is the same in all the lists (i.e. equal to the group's size).

With `TopKFinder`, we can evaluate any multi-tag query submitted by a user and efficiently solve Problem 13. Even though the inverted index itself is not original, its application to interactive recommendation systems is on of the novelties of our work.

8.7 Experiments

In this section, we present the thorough experimental evaluation that we conducted to evaluate the proposed search framework. We begin with a discussion of the datasets used throughout the section, and proceed with a detailed discussion of each experiment.

8.7.1 Datasets

The `Twitter` dataset: This is a new corpus, which we composed particularly for the purposes of this work. The corpus is built based on data collected from Twitter.com, a popular social networking platform, where one can "follow" other users and get updates on their posts. The dataset is constructed as follows: first, we obtain the list of the 1000 users in Twitter with the most followers (from TwitterHolic.com. We then crawl Twitter to retrieve the set of followers for each of these users. **The reason for focusing on the top-1000 users is that they are widely known, making the verification of our results intuitive.** Clearly, reporting groups of unknown individuals would be rather cryptic and impossible to evaluate.

After a detailed inspection of the data, we identified five entity profiles that represent the most dominant types among these highly-followed entities: *Music Artist, TV Personality, Athlete, Business Person* and *Other* (e.g. authors, bloggers, politicians). These include real-life public figures and share the following attributes: the occupation (also the name of the profile), the gender, the age group (e.g. 20-30), the country of origin, the state of origin (or city if non-usa), and the particular type or genre each person belongs to, within their bounds of their profession. This includes the music type(s) for artists, the genre(s) for TV personalities, the different properties of people assigned to the $Other$ profile (e.g. author, blogger, columnist), and the specific sport for athletes.

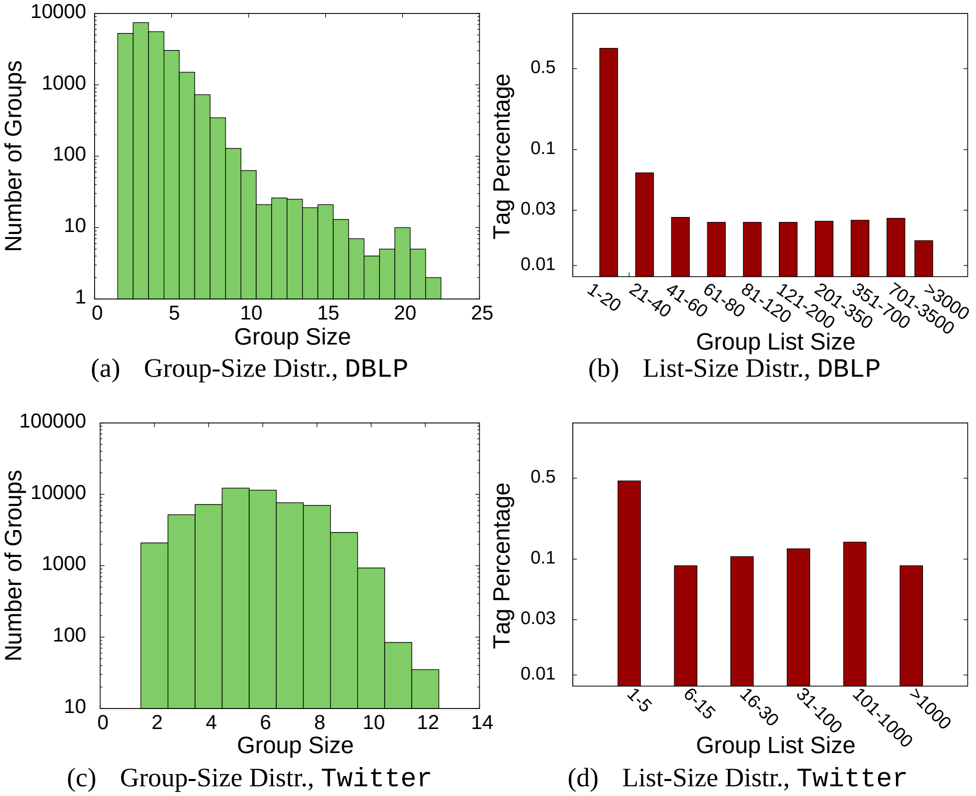
The TagSets for the followed individuals are populated in an entirely automated manner. Since TwitterHolic.com provides the actual names of the top-1000 users, we build a focused crawling and parsing system that, given a name, retrieves the required information from the Web. For TV Personalities, we use the `imdb.com` website, which hosts all the required information, including the relevant genres for each person (we only kept the top-3 genres per person, as ranked by imdb). For all other profiles we use `Wikipedia.com`, which maintains all the required profile information in a separate entry within the HTML template. The crawling system successfully retrieved the profile information for about 500 individuals. A manual examination of the unidentified entities verified that they were either not real-life people (e.g. cnn.com), spam (e.g. fake accounts), or simply users for which the information was not available on Wikipedia or imdb. The 500 profiled individuals constitute the set V of endorsed entities, in the context of a social endorsement network $G = (U, V, E)$. The set of endorsers U is represented by the entire population of followers, which consisted of $6,436,382$ distinct Twitter users (by username). The minimum number of endorsers per group (as a percentage of the total number of users) was set to 0.007.

The `DBLP` dataset: To create the second benchmark for our experiments, we use a snapshot of the data taken from the DBLP Bibliography Server on April 12, 2006[1] .

For each published paper, the snapshot contains the title, the set of authors, and the set of cited papers. Using this information, we construct our social endorsement network $G = (U, V, E)$ as follows: the set of endorsers U consists of all the papers that reference

[1] http://kdl.cs.umass.edu/data/dblp/dblp-info.html

at least one other paper. The set of endorsed entities V consists of the authors that have at least one citation to one of their papers. Thus, the set of endorsement edges E is populated by adding an edge from a paper in U to an author in V, if the paper cites the author's work. Finally, the TagSet of each author consists of the distinct (stemmed) terms that appear in his papers' titles. Alternatively, one could use the set of authors to represent both the endorsers and the endorsed entities. However, this would fail to capture cases where an author A cites multiple papers of another author B. The collection includes 456764 distinct authors and 728510 research papers (we discarded PhD and masters theses). The minimum number of endorsers per group (as a percentage of the total number of users) was set to 0.005.



(a) Group-Size Distr., `DBLP`

(b) List-Size Distr., `DBLP`

(c) Group-Size Distr., `Twitter`

(d) List-Size Distr., `Twitter`

Figure 8.4: Quantitative analysis of our framework, as applied on the `Twitter` and `DBLP` datasets. Figures 8.4(a) and 8.4(c) show the distribution of the reported groups' sizes for `DBLP` and `Twitter`, respectively. Figures and 8.4(b) and 8.4(d) show histograms of the different list sizes in the Inverted Index. For example, for `DBLP`, more than 50% of the tags were mapped to lists of at most 20 groups.

8.7.2 Quantitative Analysis

Here, we perform a detailed quantitative analysis of our framework on the `Twitter` and `DBLP` datasets.

Group Size Distribution: Figures 8.4(a) and 8.4(c) show the distribution of the various group sizes for `DBLP` and `Twitter`, respectively. The x-axis holds the different group cardinalities, while the y-axis shows the number of groups with this particular cardinality (in log-scale). For `DBLP`, the majority of the groups consist of 2-10 authors, while very few have over 20 members. For `Twitter`, the reported groups are generally smaller, with the largest groups consisting of 12 entities. This can be explained by the fact that the number of distinct entities in `Twitter` is considerably smaller (500 individuals, Vs. several thousand authors in `DBLP`), making it less likely to find large groups that share a significant number of followers and also have overlapping TagSets. In addition, the profiles in `DBLP` typically consist of numerous tags (twelve per author, on average), making it easier to identify groups of authors with overlapping TagSets.

Inverted Index: Next, we evaluate the inverted-index structure employed by our framework, by examining the size of the group-lists mapped to the indexed tags. For both datasets, a clear majority of the lists in the inverted index are small, leading to a compact structure that is easy to stored and probe.

Figures 8.4(b) and 8.4(d) show histograms of the list sizes for `DBLP` and `Twitter`, respectively. Each bar represents a size range (e.g. the first bar on Figure 8.4(b) represents all lists of size between 1 and 20). The y-axis (in log-scale) marks the percentage of tags that are mapped to a list with a size that falls within the respective range.

For `DBLP`, Figure 8.4(b) shows that over 50% of the tags where included in the TagSets of less than 20 groups. The 3 most popular tags were “databases”, “systems” and “data”, which appeared in 22326, 20015 and 19645 groups, respectively. These fall within the 2% of the tags that were mapped to more than 3000 groups.

For Twitter, Figure 8.4(d) shows that around 50% of the lists in the index contained between 1 and 5 groups. For this dataset, the 3 most popular tags where "Male", "Age[30-40]" and "TV Personality", which appeared in 48033, 2441 and 2054 groups, respectively.

8.7.3 Qualitative Analysis

Next, we evaluate the quality of our results on the Twitter and DBLP datasets. First, we create a set of 10 queries for each dataset. For DBLP, the first 9 queries are taken from the session names of the SIGKDD conference from 2006 (the same year when the data was collected). We also added a 10th query ("world wide web") for relevance. For Twitter, the queries consist of popular tags from the corpus, in order to enhance the verifiability of the results. For each query, we report the top-1 group of entities returned by our search framework, as well as the number of common endorsers per group. The results for DBLP and Twitter are shown in Tables 8.1 and 8.2, respectively.

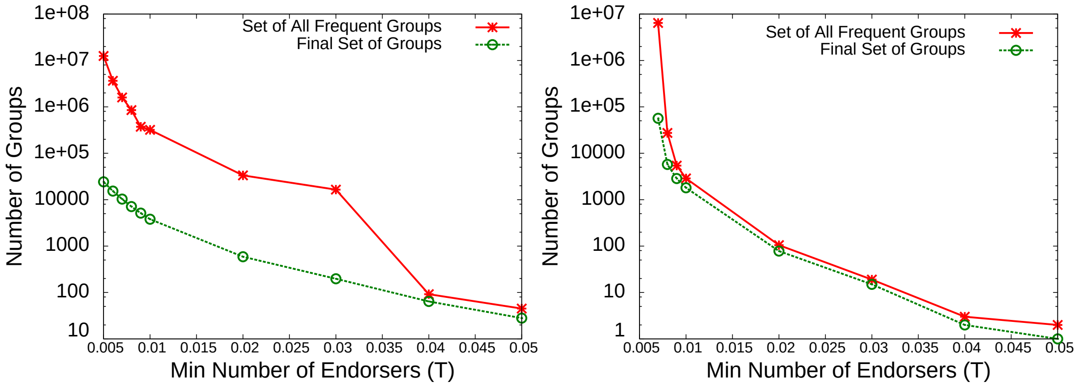
Discussion of the Results: For DBLP, we are looking for large groups of authors that have the terms of the query in their TagSets, and whose work is often cited in the same papers. As can be seen from the table, the reported groups consist of highly-cited and well-known authors. This was anticipated, since authors with numerous papers are not only more likely to be cited, but also more likely to have larger, more diverse TagSets that overlap with those of other authors. Certain names are included in the groups for many queries, indicating that the respective authors have been active in different areas, while being able to attract a significant number of citations. Another important observation is that co-authorships can be a deciding factor in the formation of groups of co-cited entities. A characteristic example is that of entry $\#9$: Won Kim, Nat Ballou, Jorge F. Garza and Darrell Woelk were all included in the top-1 group for the query "privacy", partly due to their highly-cited paper "A Distributed Object-Oriented Database System Supporting Shared and Private Databases".

For `Twitter`, we are looking for groups of individuals that match the specified query, and share a significant number of followers. As can be seen in Table 8.2, the reported groups consist of people who are well-known in their respective fields. Less focused queries lead to more diverse groups: the group reported for “Men between the ages of 30 and 40” consists of 2 actors, 1 athlete and 4 people from the business world. Interestingly enough, over 68000 Twitter users chose to follow these individuals.

Particularly interesting observations can be made from examining the groups reported for queries $\#2, \#3$ and $\#4$, which are ordered from the more general to the more focused one. For query $\#2$, a group of 4 pop-music artists is reported. Query $\#3$ is more refined, asking for groups of *female* pop-music artists. Britney Spears is the only person reported for both queries, indicating that she shares a significant number of followers with both male and female artists. Note that Britney Spears had the third largest number of followers among all the individuals in `Twitter`. The first 2 positions are held by Ashton Kutcher and Ellen DeGeneres (TV personalities), who are also included in top-1 groups. Britney Spears is also included in the top-1 group for query $\#4$. This query is even more focused, asking for women that are also between the ages of 20 and 30. An interesting observation here is that the reported group has more followers than those reported for the previous two queries, even though it is more refined. This is beacuse the group has only 3 members (while the groups for queries $\#2$ and $\#3$ had 4). As described in Section 8.6, we prefer large groups, while using the number of followers for secondary ranking.

8.7.4 Redundancy Filtering

Here, we evaluate the redundancy filter described in Section 8.5. First, we use the mechanism described in Section 8.3 to obtain the complete set of popular groups for both datasets. We repeat the experiments for different values for the minimum number of common endorsers T, expressed as a percentage of the total number of users (i.e. followers on `Twitter` and papers on `DBLP`). For each value of T, we apply the `GroupFilter`



(a) DBLP Dataset (b) Twitter Dataset

Figure 8.5: Effects of the Group Redundancy Filter for the Twitter and DBLP datasets.

algorithm and compute the number of eliminated groups. The results are shown in Figures 8.5(a) and 8.5(b). The x-axis represents the support threshold T, while the $y - axis$ represents the number of reported groups (in logarithmic scale). For each value of T, we plot the number of groups before and after the application of the redundancy filter.

The Results illustrate that, for DBLP, redundant groups cover a very high percentage of the unfiltered set. Our filter eliminates this redundancy and produces a compact and informative set. This translates to significant computational savings for a framework that needs to maintain and search the corpus of groups. For Twitter, the volume of pruned groups was reduced. This can be explained by the fact that, especially for higher values of T, the size of the unfiltered corpus was already quite small, compared to the respective number for DBLP. However, for lower values of T, the number of filtered groups was still significant. For example, for $T = 0.007$, the number of groups dropped from 6436171 to 56695, while for $T = 0.008$ it went from to 27410 to 5756.

8.8 Conclusion

In this work, we formalized the problem of *interactive recommendations in social endorsement networks*. We presented an efficient, query-driven framework for the solution of the problem, able to make high-quality and explainable recommendations. In addition, our framework is equipped with a filtering mechanism for the elimination of redundancy, which can be used reduce the size of the corpus and produce a crisp and informative dataset. The entire recommendation system is designed in a principled and efficient manner, making it ideal for large-scale systems. Finally, we illustrated the efficacy of our methods in a thorough experimental evaluation on real datasets.

Table 8.1: Groups of Authors in DBLP

	Tag-Query	**Top-1 Group of Authors**	**# Common Citations**
1.	classification	Tomasz Imielinski, Rakesh Agrawal, Sakti P. Ghosh, Balakrishna R. Iyer, Arun N. Swami	54
2.	web mining	Serge Abiteboul, Stefano Ceri	70
3.	clustering	Jiawei Han, Philip S. Yu, Rakesh Agrawal, Jong Soo Park, Arun N. Swami	43
4.	graph mining	Jiawei Han, Philip S. Yu, Rakesh Agrawal, Ming-Syan Chen	54
5.	time series	H. V. Jagadish, R. Ramakrishnan	127
6.	pattern mining	Jiawei Han, Philip S. Yu, R. Agrawal, R. Srikant, Jong Soo Park, Ming-Syan Chen	49
7.	text mining	Rakesh Agrawal, Ramakrishnan Srikant, Heikki Mannila	71
8.	structured data	Nievergelt, C. Faloutsos, H. Hinterberger, B. Seeger, J. , K. C. Sevcik, H.-P. Kriegel, A. Guttman	55
9.	privacy	Won Kim, Nat Ballou, Jorge F. Garza, Darrell Woelk	228
10.	world wide web	Alberto O. Mendelzon, Alon Y. Halevy, Anand Rajaraman, Joann J. Ordille	40

Table 8.2: Groups of Infividuals in `Twitter`

	Tag-Query	Top-1 Group of Individuals	#Followers
1.	Athlete	Shaquille O'Neal (basketball), Lance Armstrong (cycling), Tony Hawk (skateboarding)	83309
2.	Music Artist, Pop	Britney Spears, Diddy, MC Hammer, Sara Bareilles	79443
3.	Music Artist, Pop, Female	Britney Spears, Ashlee Simpson, Sara Bareilles, Maledy Moore	69056
4.	Music Artist, Pop, Female, Age[20-30]	Britney Spears, Ashlee Simpson, Lily Rose Allen	99765
5.	Business person, Age[30-40]	Evan Williams, Biz Stone, Jack Dorsey (Twitter), Michael Arrington (TechCrunch) , Kevin Rose (Digg)	71017
6.	TV Personality, Female	Ellen DeGeneres, Martha Stewart, Brooke Burke, Felicia Day, Veronica Belmont	69910
7.	TV Personality, Female, Age[50-60]	Ellen DeGeneres, Oprah Winfrey	286545
8.	Music Artist, New York	Diddy, 50 Cent, Mariah Carey	90294
9.	Comedian, New York	Jimmy Fallon, Danny Masterson	117300
10.	Male, Age[30-40]	Ashton Kutcher, Lance Armstrong, Evan Williams, Kevin Rose, Wil Wheaton, Michael Arrington, Biz Stone	68429

Bibliography

[1] T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D Gunopulos. On burstiness-aware search for document sequences. In *Proc. of SIGKDD*, pages 477–486, 2009.

[2] Theodoros Lappas and Dimitrios Gunopulos. Efficient confident search in large review corpora. In *ECML/PKDD (2)*, pages 195–210, 2010.

[3] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *KDD*, pages 467–476, 2009.

[4] Theodoros Lappas, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. Finding effectors in social networks. In *KDD*, pages 1059–1068, 2010.

[5] Theodoros Lappas and Dimitrios Gunopulos. Interactive recommendations in social endorsement networks. In *RecSys*, pages 127–134, 2010.

[6] National Digital Newspaper Program (NDNP), http://www.loc.gov/ndnp.

[7] David P. Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *J. Comput. Syst. Sci.*, 52(3):453–470, 1996.

[8] David Dobkin and David Eppstein. Computing the discrepancy. In *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, pages 47–52, New York, NY, USA, 1993. ACM.

[9] Bernard Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, New York, NY, USA, 2000.

[10] Deepak Agarwal, Jeff M. Phillips, and Suresh Venkatasubramanian. The hunting of the bump: on maximizing statistical discrepancy. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1137–1146, New York, NY, USA, 2006. ACM.

[11] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[12] Jon Kleinberg. Bursty and hierarchical structure in streams. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 91–101, New York, NY, USA, 2002. ACM.

[13] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S. Yu, and Hongjun Lu. Parameter free bursty events detection in text streams. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 181–192. VLDB Endowment, 2005.

[14] Qi He, Kuiyu Chang, and Ee-Peng Lim. Analyzing feature trajectories for event detection. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 207–214, New York, NY, USA, 2007. ACM.

[15] Michail Vlachos, Christopher Meek, Zografoula Vagena, and Dimitrios Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 131–142, New York, NY, USA, 2004. ACM.

[16] Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345, New York, NY, USA, 2003. ACM.

[17] Qi He, Kuiyu Chang, Ee-Peng Lim, and Jun Zhang. Bursty feature representation for clustering text streams. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 491–496, 2007.

[18] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. On the bursty evolution of blogspace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 568–576, New York, NY, USA, 2003. ACM.

[19] Qi He and Kuiyu Chang and Ee-Peng Lim. Using burstiness to improve clustering of topics in news streams. In *ICDM '07: Proceedings of the International Conference on Data Mining*, Washington, DC, USA, 2007. IEEE Computer Society.

[20] N. Bansal and N. Koudas. Blogscope: a system for online analysis of high volume text streams. *VLDB*, 2007.

[21] N. Bansal and N. Koudas. Blogscope: spatio-temporal analysis of the blogosphere. *WWW*, 2007.

[22] W.-C. Tien C.-H. Cheng, K.-Y. Chen and K.-M. Chao. Improved algorithmms for the k maximum-sums problems. *Theor. Comput. Sci.*, pages 162–170, 2006.

[23] Walter L. Ruzzo and Martin Tompa. A linear time algorithm for finding all maximal scoring subsequences. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 234–241. AAAI Press, 1999.

[24] R. L. Rivest T. H. Cormen, C. E. Leiserson and C. Stein. Bump hunting in high-dimensional data. *Introduction to Algorithms, Second Edition*, September 2001.

[25] Theodoros Lappas, Benjamin Arai, Manolis Platakis, Dimitrios Kotsakos, and Dimitrios Gunopulos. On burstiness-aware search for document sequences. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 477–486, 2009.

[26] Jon Kleinberg. Bursty and hierarchical structure in streams. *Data Min. Knowl. Discov.*, 7:373–397, October 2003.

[27] Michael Mathioudakis, Nilesh Bansal, and Nick Koudas. Identifying, attributing and describing spatial bursts. *Proceedings of Very Large Data Bases (VLDB) Conference*, pages 1091–1102, 2010.

[28] Egon Balas, Vašek Chvátal, and Jaroslav Nešetřil. On the maximum weight clique problem. *Math. Oper. Res.*, 12:522–535, 1987.

[29] U. I. Gupta, D. T. Lee, and J. Y.-T Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12(4):459–467, 1982.

[30] Chi-Su Wang and Ruay Shiung Chang. A parallel maximal cliques algorithm for interval graphs with applications. *J. Inf. Sci. Eng.*, 13(4):649–663, 1997.

[31] David P. Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *J. Comput. Syst. Sci.*, 52:453–470, June 1996.

[32] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS '01*.

[33] http://www.topix.com.

[34] T. Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, 22(176):88–93, 1975.

[35] http://www.wikipedia.org.

[36] Vivek K. Singh, Mingyan Gao, and Ramesh Jain. Situation detection and control using spatio-temporal analysis of microblogs. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 1181–1182, 2010.

[37] Angelo Dalli. System for spatio-temporal analysis of online news and blogs. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 929–930, 2006.

[38] Yu Meng and Margaret H. Dunham. Efficient mining of emerging events in a dynamic spatiotemporal environment. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 750–754, 2006.

[39] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: news in tweets. pages 42–51, 2009.

[40] Qiaozhu Mei, Chao Liu, Hang Su, and ChengXiang Zhai. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 533–542, 2006.

[41] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 851–860, 2010.

[42] Nilesh Bansal and Nick Koudas. Blogscope: spatio-temporal analysis of the blogosphere. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 1269–1270, 2007.

[43] Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, and Xing Xie. Searching trajectories by locations: an efficiency study. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 255–266, 2010.

[44] Marios Hadjieleftheriou, George Kollios, Vassilis J. Tsotras, and Dimitrios Gunopulos. Indexing spatiotemporal archives. 15(2):143–164, 2006.

[45] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. pages 1068–1080, 2008.

[46] Sangkyum Kim, Xin Jin, and Jiawei Han. Sparclus: Spatial relationship pattern-based hierarchial clustering. In *SDM*, pages 49–60, 2008.

[47] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. *TraClass*: trajectory classification using hierarchical region-based and trajectory-based clustering. 1(1):1081–1094, 2008.

[48] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. 3(1):723–734, 2010.

[49] Nikos Mamoulis, Huiping Cao, George Kollios, Marios Hadjieleftheriou, Yufei Tao, and David W. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 236–245, 2004.

[50] Jimeng Sun, Dimitris Papadias, Yufei Tao, and Bin Liu. Querying about the past, the present, and the future in spatio-temporal. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 202–213, 2004.

[51] Marcos R. Vieira, Petko Bakalov, and Vassilis J. Tsotras. On-line discovery of flock patterns in spatio-temporal data. pages 286–295, 2009.

[52] Marcos R. Vieira, Petko Bakalov, and Vassilis J. Tsotras. Querying trajectories using flexible patterns. In *International Conference on Extending Database Technology (EDBT)*, pages 406–417, 2010.

[53] Hui Yang, Srinivasan Parthasarathy, and Sameep Mehta. A generalized framework for mining spatio-temporal patterns in scientific data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 716–721, 2005.

[54] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 791–800, 2009.

[55] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *ICDE*, 2001.

[56] Zhu Zhang and Balaji Varadarajan. Utility scoring of product reviews. In *CIKM*, 2006.

[57] Soo min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *EMNLP '06*.

[58] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *WSDM '08*.

[59] Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. Low-quality product review detection in opinion summarization. In *(EMNLP-CoNLL)*, 2007.

[60] N. Archak, A. Ghose, and P. Ipeirotis. Show me the money! Deriving the pricing power of product features by mining consumer reviews. In *SIGKDD*, 2007.

[61] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *WWW '03*, 2003.

[62] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *SIGKDD*, 2004.

[63] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, 2005.

[64] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *(EMNLP)*, 2002.

[65] Peter D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL*, 2002.

[66] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *AAAI*, 2004.

[67] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *HLT '05:*.

[68] R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano. Text mining for product attribute extraction. *SIGKDD Explorations Newsletter*, 2006.

[69] Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. Feature subsumption for opinion analysis. In *EMNLP*, 2006.

[70] Li Zhuang, Feng Jing, Xiao-yan Zhu, and Lei Zhang. Movie review mining and summarization. In *CIKM*, 2006.

[71] Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, 2006.

[72] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 2005.

[73] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. Skyline with presorting. *ICDE*, 2003.

[74] Alberto Caprara, Matteo Fischetti, and Paolo Toth. Algorithms for the set covering problem. *Annals of Operations Research*, 1996.

[75] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 1979.

[76] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB '98*.

[77] T. Bell. Extensive reading: speed and comprehension. In *The Reading Matrix, 1(1)*, 2001.

[78] M.D. Lee and M. Welsh. An empirical evaluation of models of text document similarity. In *CogSci*, 2005.

[79] R.K. Ando. Latent semantic space: Iterative scaling improves precision of inter-document similarity measurement. In *Proc. of SIGIR*, pages 216–223, 2000.

[80] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[81] Charles L.A. Clarke. Novelty and diversity in information retrieval evaluation. In *Proc. of ACM SIGIR*, pages 659–666, 2008.

[82] W.H DuBay. The principles of readability. In *Impact Information*, 2004.

[83] J.S. Chall and E. Dale. *Readability revisited: The New Dale-Chall Readability Formula*. Brookline Books, 1995.

[84] K. Collins-Thompson and J. Callan. A language modeling approach to predicting reading difficulty. In *Proc. of HLT/NAACL*, 2004.

[85] R.F. Flesch. A new readability yardstick. In *J Appl Psychol 32*, pages 221–224, 1948.

[86] A.J. Stenner. Measuring reading comprehension with the lexile framework. In *American Conference on Adolescent/Adult Literacy*, 1996.

[87] M. Heilman, K. Collins-Thompson, and M. Eskenazi. An analysis of statistical models and features for reading difficulty prediction. In *Proc. of Workshop on innovative Use of NLP For Building Educational Applications*, pages 71–79, 2008.

[88] S. E. Petersen and M. Ostendorf. A machine learning approach to reading level assessment. In *Comput. Speech Lang. 23(1)*, pages 89–106, 2009.

[89] K. Collins-Thompson and J. Callan. Predicting reading difficulty with statistical language models. In *JASIST '05*.

[90] E. Pitler and A. Nenkova. Revisiting readability: a unified framework for predicting text quality. In *EMNLP '08*.

[91] Niels Ott. Information retrieval for language learning: An exploration of text difficulty measures. In *Master's Thesis in Computational Linguistics, Universitt Tbingen*, 2009.

[92] Alexandra L. Uitdenbogerd. Web readability and computer-assisted language learning. In *Proc. of Australasian Language Technology Workshop*, pages 99–106, 2006.

[93] Brian M. Friel and Shelia M. Kennison. Identifying German-English cognates, false cognates, and non-cognates: methodological issues and descriptive norms. In *Bilingualism: Language and Cognition 4(3)*, pages 249–274, 2001.

[94] Stefan Schulz, Kornél Markó, Eduardo Sbrissia, Percy Nohama, and Udo Hahn. Cognate mapping - a heuristic strategy for the semi-supervised acquisition of a spanish lexicon from a portuguese seed lexicon. In *COLING '04*.

[95] R. Mitkov, V. Pekar, D. Blagoev, and A. Mulloni. Methods for extracting and classifying pairs of cognates and false friends. In *Machine Translation 21 (1)*, pages 29–53, 2007.

[96] S. E. Petersen. Natural language processing tools for reading level assessment and text simplification for bilingual education. In *Doctoral Thesis, University of Washington.*, 2007.

[97] Lucius Adelno Sherman. *Analytics Of Literature: A Manual For The Objective Study Of English Prose And Poetry*. Ginn and Company, 1893.

[98] J.P. Kincaid, R.P Fishburne, R.L. Rogers, and B.S. Chissom. Derivation of new readability formulas for navy enlisted personnel. In *Research Branch Report 8-75, Millington, TN: Naval Technical Training*, 1975.

[99] G.K. Zipf. *The Psychology of Language*. Houghton Mifflin, 1935.

[100] Toni Amstad. Wie verstndlich sind unsere zeitungen? In *Universitt Zrich: Dissertation*, 1978.

[101] T. Brants, A.C. Popat, P. Xu, F.J. Och, and J. Dean. Large language models in machine translation. In *Proc. of EMNLP-CoNLL*, pages 858–867, 2007.

[102] Job Schepens. Distributions of cognates in europe based on the levenshtein distance. In *Bachelor Thesis*, 2008.

[103] Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. German decompounding in a difficult corpus. In *Computational Linguistics and Intelligent Text Processing*, pages 128–139, 2008.

[104] Anne Schiller. German compound analysis with wfsc. In *Finite-State Methods and Natural Language Processing*, 2006.

[105] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[106] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. An optimal and progressive algorithm for skyline queries. In *Proc. of ACM SIGMOD*, pages 467–478, 2003.

[107] Chia-Jung Lee, Chin-Hui Chen, Shao-Hang Kao, and Pu-Jen Cheng. To translate or not to translate? In *Proc. of SIGIR*, pages 651–658, 2010.

[108] Adil Baykasoglu, Turkay Dereli, and Sena Das. Project team selection using fuzzy optimization approach. *Cybern. Syst.*, 38(2):155–185, 2007.

[109] Shi-Jie Chen and Li Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Transactions on Engineering Management*, 51(2):111–124, 2004.

[110] Armen Zzkarian and Andrew Kusiak. Forming teams: an analytical approach. *IIE Transactions*, 31:85–97, 2004.

[111] Hyeongon Wi, Seungjin Oh, Jungtae Mun, and Mooyoung Jung. A team formation model based on knowledge and collaboration. *Expert Syst. Appl.*, 36(5):9121–9134, 2009.

[112] Erin L. Fitzpatrick and Ronald G. Askin. Forming effective worker teams with multi-functional skill requirements. *Comput. Ind. Eng.*, 48(3):593–608, 2005.

[113] M. Gaston, J. Simmons, and M. desJardins. Adapting network structures for efficient team formation. In *In Proceedings of the AAAI Fall Symposium on Artificial Multiagent Learning*, 2004.

[114] Michelle Cheatham and Kevin Cleereman. Application of social network analysis to collaborative team formation. In *CTS '06: Proceedings of the International Symposium on Collaborative Technologies and Systems*, pages 306–311.

[115] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, New York, NY, USA, 2006. ACM.

[116] Matthew Jackson. Network formation. *The New Palgrave Dictionary of Economics and the Law*, 2008.

[117] Esther M. Arkin and Refael Hassin. Minimum-diameter covering problems. *Networks*, 36(3):147–155, 2000.

[118] G. Reich and P. Widmayer. Beyond steiner's problem: a VLSI oriented generalization. In *Proceedings of the fifteenth international workshop on Graph-theoretic concepts in computer science*, pages 196–210, 1990.

[119] Chandra Chekuri, Guy Even, and Guy Kortsarz. A greedy approximation algorithm for the group steiner problem. *Discrete Appl. Math.*, 154(1):15–34, 2006.

[120] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge UP, 2004.

[121] Vijay Vazirani. *Approximation Algorithms*. Springer, 2003.

[122] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.

[123] C. W. Duin, A. Volgenant, and S. Vo[ss]. Solving group Steiner problems as Steiner problems. *European Journal of Operational Research*, 154(1):323–329, 2004.

[124] Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.

[125] Daniel Gruhl, David Liben-Nowell, Ramanathan V. Guha, and Andrew Tomkins. Information diffusion through blogspace. *SIGKDD Explorations*, 6(2):43–52, 2004.

[126] Nitin Agrawal, Huan Liu, Lei Tang, and Philip S. Yu. Identifying the influential bloggers in a community. In *WSDM*, 2008.

[127] Michael Mathioudakis and Nick Koudas. Efficient identification of starters and followers in social media. In *EDBT*, pages 708–719, 2009.

[128] Paat Rusmevichientong, Shenghuo Zhu, and David Selinger. Identifying early buyers from purchase data. In *KDD*, pages 671–677, 2004.

[129] Shishir Bharathi, David Kempe, and Mahyar Salek. Competitive influence maximization in social networks. In *WINE*, pages 306–311, 2007.

[130] Pedro Domingos and Matthew Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

[131] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[132] Heikki Mannila and Evimaria Terzi. Finding links and initiators: a graph-reconstruction problem. In *SDM*, pages 1207–1217, 2009.

[133] James Aspnes, Kevin L. Chang, and Aleksandr Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *J. Comput. Syst. Sci.*, 72(6), 2006.

[134] Deepayan Chakrabarti, Yang Wang 0008, Chenxi Wang, Jure Leskovec, and Christos Faloutsos. Epidemic thresholds in real networks. *ACM Trans. Inf. Syst. Secur.*, 10(4), 2008.

[135] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[136] Mikko Koivisto. Advances in exact bayesian structure discovery in bayesian networks. In *UAI*, 2006.

[137] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211–223, 2001.

[138] J. Goldenberg, B. Libai, and E. Muller. Using complex systems analysis to advance marketing theory development. *Academy of Marketing Science Review*, 2001.

[139] Ronald Fagin, Ramanathan V. Guha, Ravi Kumar, Jasmine Novak, D. Sivakumar, and Andrew Tomkins. Multi-structural databases. In *PODS*, pages 184–195, 2005.

[140] Ravi Kumar, Kunal Punera, and Andrew Tomkins. Hierarchical topic segmentation of websites. In *KDD*, pages 257–266, 2006.

[141] A. Tamir. An $o(pn^2)$ algorithm for the p-medial and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.

[142] Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997.

[143] Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and ming Li. Approximation algorithms for directed steiner problems. *J. Algorithms*, 33:73–91, 1999.

[144] Y.J. Chu and T.H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

[145] J. Edmonds. Optimum branchings. *J. Research of the National Bureau of Standards*, 71B:233–240, 1967.

[146] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *ICDEW ’07*.

[147] Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. The slashdot zoo: mining a social network with negative edges. In *WWW ’09*, NY, USA.

[148] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *WWW ’10*.

[149] Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *RecSys ’09*.

[150] P. Bonhard and M. A. Sasse. ’knowing me, knowing you’ – using profiles and social networking to improve recommender systems. *BT Technology Journal*, 2006.

[151] Weiyang Lin, Sergio A. Alvarez, and Carolina Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Min. Knowl. Discov.*, 2002.

[152] J. J. Sandvig, Bamshad Mobasher, and Robin Burke. Robustness of collaborative recommendation based on association rule mining. In *RecSys ’07*.

[153] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Effective personalization based on association rule discovery from web usage data. In *WIDM ’01*.

[154] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *RecSys ’09*.

[155] Paul Alexandru Chirita, Stefania Costache, Wolfgang Nejdl, and Siegfried Handschuh. P-tag: large scale automatic generation of personalized annotation tags for the web. In *WWW ’07*.

[156] Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C. Lee Giles. Real-time automatic tag recommendation. In *SIGIR ’08*.

[157] Nikhil Garg and Ingmar Weber. Personalized, interactive tag recommendation for flickr. In *RecSys ’08*.

[158] Valentina Zanardi and Licia Capra. Social ranking: uncovering relevant content using tag-based recommender systems. In *RecSys ’08*.

[159] Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, and Pierpaolo Basile. Integrating tags in a semantic content-based recommender. In *RecSys ’08*.

[160] Shilad Sen, Jesse Vig, and John Riedl. Tagommenders: connecting users to items through tags. In *WWW ’09*.

[161] Aleksandra Klasnja Milicevic, Alexandros Nanopoulos, and Mirjana Ivanovic. Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions. *Artif. Intell. Rev.*, 2010.

[162] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *RecSys '08*.

[163] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis. 2010.

[164] Paolo Viappiani, Pearl Pu, and Boi Faltings. Conversational recommenders with adaptive suggestions. In *RecSys '07*, 2007.

[165] Derek Bridge and Francesco Ricci. Supporting product selection with query editing recommendations. In *RecSys '07*.

[166] Ralf Schenkel, Tom Crecelius, Mouna Kacimi, Sebastian Michel, Thomas Neumann, Josiane X. Parreira, and Gerhard Weikum. Efficient top-k querying over social-tagging networks. In *SIGIR'08*.

[167] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 2004.

[168] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. In *The Adaptive Web: Methods and Strategies of Web Personalization*. 2007.

[169] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings*.

[170] Yi Zhen, Wu-Jun Li, and Dit-Yan Yeung. Tagicofi: tag informed collaborative filtering. In *RecSys '09*.

[171] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. pages 487–499, 1994.

[172] Jrg Hoffmann and Jana Koehler. A new method to index and query sets. In *IJCAI99*.