Spatial, Temporal, and Textual Retrieval and Analysis of Geotagged Posts

Paras Mehta

Fachbereich Mathematik und Informatik
Freie Universität Berlin

Dissertation zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

Berlin 2017

Gutachter: Prof. Agnès Voisard, Ph.D.
Institut für Informatik
Freie Universität Berlin
Takustr. 9
14195 Berlin
Germany
agnes.voisard@fu-berlin.de
http://page.mi.fu-berlin.de/voisard/

Prof. Dieter Pfoser, Ph.D.
Department of Geography and Geoinformation Science
George Mason University
4400 University Drive, MS 6C3
Fairfax, VA, 22032
United States
dpfoser@gmu.edu
https://cos.gmu.edu/ggs/people/faculty-staff/dieter-pfoser/

Tag der Disputation: 20.12.2017

To Judith, Elias, and Amalia.

Acknowledgements

From its inception until its conclusion, this thesis has received the support and encouragement of others. Here, I would like to express my gratitude to the following individuals and organizations:

- my supervisor, Prof. Agnès Voisard, for her supervision and support throughout the PhD period. She has devoted an immense amount of time and effort in always providing me with constructive feedback and ideas, while giving me the freedom to pursue my own interests. I thank her sincerely for her unwavering patience and understanding.
- Prof. Dieter Pfoser for co-reviewing this thesis, and for his guidance and encouragement during the GEOCROWD project. I also thank him for organizing GEOCROWD, which was a stepping stone in my research career, giving me the opportunity to meet and learn from leading researchers like him.
- Dr. Dimitrios Skoutas for a very fruitful research collaboration and for his valuable pieces of advice during the PhD. I am also very grateful to him for his in-depth proofreading of this thesis. A special vote of thanks also to Dr. Dimitris Sacharidis and Dr. Kostas Patroumpas, with whom I had the opportunity to collaborate successfully on research problems. I always found our Skype conversations very insightful and interesting.
- Prof. Heinz Schweppe for always showing interest in my work and for supplying me with very helpful feedback on my research. I also offer him my sincerest gratitude for proofreading this thesis in detail.
- present and former colleagues at the Databases and Information Systems group, who were the first friends I made after moving to Berlin. I will fondly remember our countless interesting discussions over lunch and the friendly atmosphere at work. In particular, I would like to thank Dr. Jürgen Broß, Dr. Sebastian Müller, Daniel Kreßner, To Tu Cuong, Tobias Albig, and Nicolas Lehmann. A special thanks to Nicolas Lehmann for his cooperation in the City.Risks project and for his support with organizational tasks.
- the very motivated students with whom I had the chance to work on some very innovative projects, and enjoy many interesting discussions and brainstorming sessions. Specifically, I would like to thank Marc Simons, Manuel Kotlarski, Christian Windolf, Erik Zocher, Luisa Castaño, Andras Komaromy, and Kadir Tugan.
- Heike Eckart for her constant support with administrative activities and for even accompanying me to the Foreigners' Office.

- the EU Marie Curie Initial Training Network GEOCROWD and the EU H2020 project City.Risks for funding this research.
- my partner, Judith Schenkel, whose love and faith have been with me through thick and thin. Without her, this thesis would not have been possible.
- my parents, Bharti and Vinod Mehta, who always wish the best for me and whose values guide me to this day.

Abstract

The proliferation of GPS-equipped mobile devices, as well as online social networks, has led to the creation of increasingly large volumes of *spatio-textual data*, i.e., data containing spatial and textual information, such as geotagged messages on Twitter and reviews for restaurants on Foursquare. Similarly, a growing amount of Internet searches now carry a spatial intent. From looking up nearby grocery stores to searching for local news, we increasingly use the Internet to find local information. Due to these factors, queries combining spatial and textual predicates, termed *spatial keyword queries*, have been studied extensively over the past few years.

Different types of spatial keyword queries have been studied in the literature, ranging from the simplest that retrieve the top-*k* relevant objects to more complex variants that identify groups of objects jointly satisfying the query. Still, the majority of existing research focuses mainly on *static* settings, such as searching for information about places. In contrast, social networks are a *dynamic* source of *crowdsourced* spatio-textual data in the form of geotagged posts (e.g., tweets, check-ins) made by users, which is being produced in *large amounts* and is evolving continuously. These characteristics of geotagged posts create several new opportunities and challenges, and call for the enhancement of existing techniques to handle this type of data.

Thus, in this thesis, we present novel techniques for the retrieval and analysis of geotagged posts. Initially, since posts consist of not only spatial and textual attributes, but also temporal information, we extend spatio-textual access methods to support spatial-temporal-textual filtering of trajectories generated via social networks. Following this, considering that the number of results found by this plain filtering can be quite high, and thus overwhelming for users, we propose a new method for identifying a small set of representative posts for a given spatial-temporal-textual filter, to allow spatio-temporal exploration of the large number of relevant posts. Nevertheless, these results can quickly become outdated with time as fresh posts are made. Thus, in our subsequent analysis, we propose methods for continuously maintaining a concise summary of a stream of posts within a sliding window, and updating the summary dynamically as the window slides. Finally, given their crowdsourced nature, geotagged posts are a rich source of people's local knowledge and opinions, which we exploit by inferring two types of patterns. First, we develop a system for the discovery and exploration of local hotspots of certain keywords, termed *locally trending topics*. In the second, we use the digital trails generated by mobile users posting on social networks for mining thematic associations among groups of locations.

keywords: spatial keyword search, spatio-temporal queries, social networks, geographic information retrieval, query processing, indexing, algorithms

Table of contents

List of figures

List of tables

CHAPTER 1

INTRODUCTION

1.1 Motivation

Five-star hotels near Berlin Central station

Movie theaters near me screening La La Land

Restaurants in Berlin city center serving Schnitzel and Apple Strudel

At a first glance, these phrases do not seem to have anything in common. However, a closer look might reveal that they are all examples of *searches for local information*. Also known as *spatial keyword queries* or *spatio-textual queries*, these searches are frequently performed using mobile devices and are in contrast to other queries, such as "*Books on leadership*", that do not have a local intent. In recent years, spatial keyword queries have assumed an increasingly important role in people's everyday lives. This can be partly attributed to the rapid surge in the use of mobile devices, such as smartphones and tablets. According to the market research firm Gartner, the sale of smartphones alone worldwide was expected to reach 1.5 billion units in 2016[1]. Using increasingly pervasive and precise positioning techniques based on GPS, WiFi, and other outdoor and indoor positioning technologies, mobile devices are now able to provide the location of the user at most times. Searching for local information has thus become very convenient and is now one of the most common activities on mobile devices[2]. As a result, **more and more online search requests are acquiring a spatial intent**. This has facilitated the rise of several location-based search providers, such as Foursquare and Yelp, that allow people to look for Points

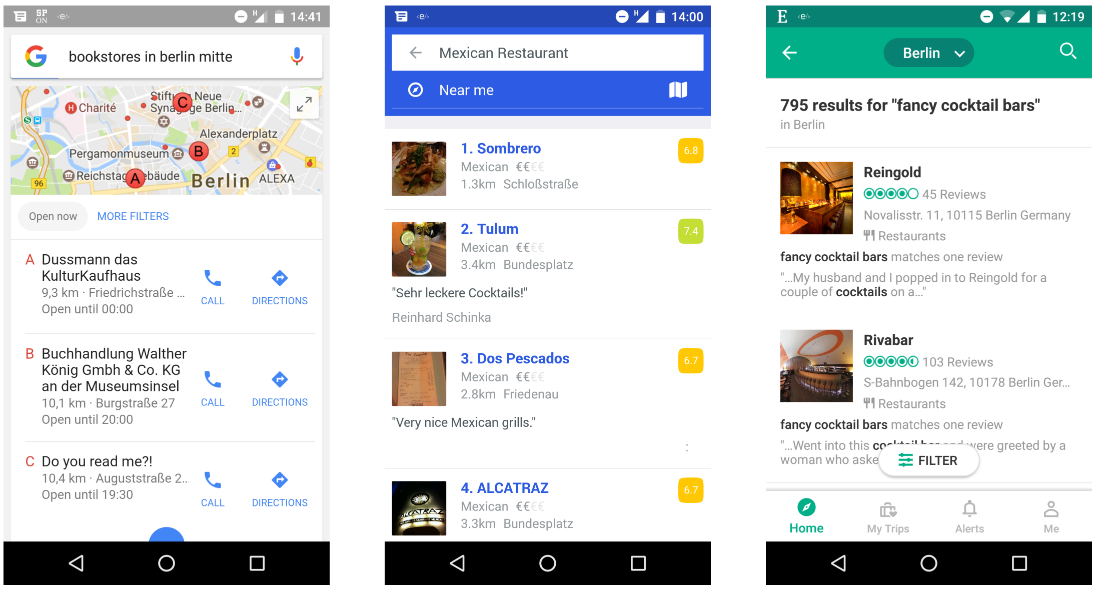
[1]http://www.gartner.com/newsroom/id/3339019
[2]https://goo.gl/7AxJ5Q

of Interest (POIs) and view ratings provided by other users. Moreover, existing search engines, such as Google and Microsoft Bing, now also support local search with a large portion of requests being generated via mobile devices[3]. Already in 2015, in 10 countries including the US and Japan, the volume of mobile search requests had exceeded that of desktop requests on Google[3], with location-related mobile searches growing 50% faster than all mobile searches[4]. Not surprisingly, major Internet companies, including Google and Facebook, have already adopted a mobile-first strategy offering users information about businesses, events, news, and friends in their area in return for personalized location-based advertisements.

Along with the proliferation of mobile devices, another major trend in the recent years has been the rapid growth of online social networks, such as Facebook and Twitter. In fact, a large portion of users with mobile devices use these to access online social networks. For example, out of Facebook's nearly 1.79 billion monthly active users, more than 1 billion access the service solely through their mobile devices[5]. By posting actively about their activities, surroundings, and opinions on social networks, ordinary users have transformed from being sole consumers into both generators and consumers of data. As a result, due to the widespread use of GPS-equipped mobile devices and social networks, there has been an **explosion in the amount of data with spatial and textual attributes** on the Web.

In light of these developments, to support efficient location-based search, there has been a **significant amount of research done recently in the area of spatial keyword queries**. Spatial keyword queries enable the retrieval of objects based on the spatial and textual predicates provided by the user in the search. For example, in a search for a local restaurant, usually the spatial part consists of a location (e.g., the current location reported by the user's mobile device) or a region of interest, (e.g., "*Berlin city center*"), whereas the textual component contains some keywords describing the user's information needs, e.g., the type of restaurant or food. The query response shows a list of *spatio-textual objects*, e.g., web pages of restaurants, that can be viewed in the order of their distance from the query location, their relevance to query keywords, their ratings, etc., or a combination of these. Each spatio-textual object is associated with a location and a set of keywords. Web pages of POIs, such as restaurants, coffee shops, and monuments, geotagged photos with descriptive tags, reviews posted about places on travel websites (e.g., TripAdvisor), and geotagged tweets are classic examples of spatio-textual objects. For example, Figure 1.1 shows

[3]https://goo.gl/a0tmab
[4]https://goo.gl/jJUfcA
[5]http://mashable.com/2016/11/02/facebook-mobile-only-users


bookstores in berlin mitte

Open now MORE FILTERS

A Dussmann das KulturKaufhaus
9,3 km · Friedrichstraße ...
Open until 00:00

B Buchhandlung Walther König Gmbh & Co. KG an der Museumsinsel
10,1 km · Burgstraße 27
Open until 20:00

C Do you read me?!
10,4 km · Auguststraße 2...
Open until 19:30

(a) Google

Mexican Restaurant

Near me

1. Sombrero 6.8
Mexican €€
1.3km Schloßstraße

2. Tulum 7.4
Mexican €€
3.4km Bundesplatz
"Sehr leckere Cocktails!"
Reinhard Schinka

3. Dos Pescados 6.7
Mexican €€
2.8km Friedenau
"Very nice Mexican grills."

4. ALCATRAZ 6.7
Mexican €€
3.3km Bundesplatz

(b) Foursquare

Berlin

795 results for "fancy cocktail bars"
in Berlin

Reingold
45 Reviews
Novalisstr. 11, 10115 Berlin Germany
Restaurants
fancy cocktail bars matches one review
"...My husband and I popped in to Reingold for a couple of cocktails on a..."

Rivabar
103 Reviews
S-Bahnbogen 142, 10178 Berlin Ger...
Restaurants
fancy cocktail bars matches one review
"...Went into this cocktail bar and were greeted by a woman who aske...

(c) TripAdvisor

Fig. 1.1 Examples of POI searches in location-based search applications.

the user interface for POI search on mobile applications of three major location-based search providers, namely Google, Foursquare, and TripAdvisor. Users can enter a set of search terms, and optionally a location; the results are a list of places whose category, description, and reviews are relevant to the search keywords and which are located close to the query location. Each entry in the results consists of some information about the place, such as the name, description, ratings, and reviews.

The main focus of research on spatial keyword queries has been on combining spatial queries with keyword search, and finding data by specifying a spatial and a textual filter. In its simplest form, a query typically comprises a region or a location and a set of keywords, and seeks all or top-k locations that contain one or more query keywords and lie within the query region or close to the query location, respectively [32]. These are termed *standard spatial keyword queries* in literature [22, 39, 42]. Characteristic examples of standard queries are searches for POIs, such as restaurants and coffee shops, that specify a location or a region of interest and some keywords to describe the desired place. To efficiently evaluate these queries, there has already been a lot of work on combining indexes for spatial search and text retrieval [32]. In addition to this, several other query variants have been studied in the literature. For example, *Collective Spatial Keyword (CSK) queries* [174, 21, 77, 24, 70] return groups of objects that together contain all the query keywords and are also close to each other, instead of the individual objects themselves. For instance, consider a tourist in a city who would like to go shopping, running, and dining. Her requirements might be better met by a group of locations, rather than a single location [21]. Similarly, other types, such as the *Prestige-Aware query* [20] and the *Preference-Aware query* [154], give higher importance to objects located close to several other relevant objects. These are motivated by the observation that people often prefer to visit a location with many relevant locations (e.g., restaurants or shops) nearby, over those with fewer relevant locations in proximity [20]. Another line of related work deals with retrieving entire *streets* [146] or *regions* [23, 61, 62] containing many relevant POIs to facilitate user exploration. A typical example here would be a search for an area with many restaurants in Berlin [23]. Various other forms of queries have been studied; for a detailed survey of existing work, see Chapter 2. Nevertheless, based on this brief discussion, it is evident that although several different types of queries have been examined, the main focus of existing research has been on the retrieval of POIs or, more generally, *static* documents associated to locations. On the other hand, in addition to the growing quantity, **diverse types of dynamic spatio-textual data are being generated by users on social networks**, making new kinds of searches and analyses possible. For example, on websites, such as Twitter and Flickr, people tend to post messages and photos about their activities, whereas on Foursquare and Yelp, users are allowed to *'check-in'* into venues and post comments and ratings.

Therefore, in this thesis, we advance the state-of-the-art in spatial keyword query processing by studying methods for the retrieval and analysis of *geotagged posts*, such as geotagged tweets, geotagged photos, and check-ins, made by users on social networks. In contrast to the typically static data, such as POIs, used in existing research, time is an important attribute in this type of information, which is ignored in classic spatio-textual query processing. Here, the textual content of a post is a short text or a set of tags, the spatial content is its geolocation, and the temporal content refers to the time of the post. Moreover, geotagged posts are being generated in large amounts continuously on social networks. This creates additional challenges and opportunities not only for analyzing and retrieving this information, but also for effectively presenting it to users. These have not been adequately addressed in existing research. Furthermore, although individual posts themselves might carry limited information, collectively they serve as a rich source of crowdsourced intelligence in the form of local opinions and knowledge about places, which can be examined to reveal insights for improving location-based services. Thus, the availability of massive volumes of geotagged posts calls for the enhancement of existing techniques to meet these challenges.

1.2 Goal

The main goal of this thesis is to present novel search techniques and solutions for supporting spatial, temporal, and textual retrieval and analysis of geotagged posts. To achieve this, we study the following problems. First, to take advantage of the additional available temporal information in posts, we address the problem of extending existing spatio-textual access methods to support temporal data. Specifically, we begin by focusing on the spatial-temporal-textual filtering of trajectories of moving objects generated by mobile users posting on social networks and by movement tracking applications. However, given the large number of posts made on social networks, the number of results produced by this plain boolean range filtering can be very high, and thus overwhelming for users. As a result, in our subsequent analysis, instead of returning all results lying within the range, we focus on finding a small diverse set of k representative posts for a given spatio-temporal range and keyword filter. The results returned can serve as seeds for spatio-temporal exploration of the large amount of relevant posts, making this technique suitable for the analysis of events and topics with large spatio-temporal footprints. Nevertheless, given that new messages are being posted constantly on social networks, the current result set can quickly become outdated with the passage of time. Hence, an important enhancement of this method is to update the results as fresh posts arrive. This is precisely the goal of our next step, where we devise techniques for generating a concise and up-to-date *summary* of posts lying within a sliding window over a stream, and updating it dynamically as the window slides. Finally, motivated by the observation that posts made at a certain location may indicate something about the location, we study the use of posts as sources of local knowledge for enriching locations and inferring patterns. We achieve this in two different ways. The first is by developing a system for detecting and exploring hotspots for a certain set of keywords, i.e., areas where posts with those keywords occur more frequently, termed *locally trending topics* in literature. The other is by leveraging users' mobility patterns and their semantic characterization of locations from social networks as evidence to identify places that are thematically associated. In the following, we outline the goal of each of these tasks.

1.2.1 Spatial-Temporal-Textual Filtering of Trajectories

The vast amount of data in the form of geotagged tweets, photos, or check-ins posted constantly on online social networks using GPS-enabled mobile devices consists of not only spatial and textual attributes, but also of temporal information. Hence, the goal of this part of our work is to extend spatio-textual retrieval methods to handle temporal data. Concretely, we address the problem of efficient evaluation of queries that perform spatial, temporal, and keyword-based filtering on historical movement data of objects that is additionally associated with textual information in the form of keywords, potentially changing at each timestamp and location. This data is available in the form of trails of mobile users posting geotagged photos or tweets and as tracking data of vehicles, ships, and animals consisting of GPS locations with textual status updates. Thus, each point in the trajectory is characterized by a location, a timestamp, and a set of tags or keywords. Consequently, we aim to evaluate queries, such as "*retrieve all users who have been in the city center of Berlin in the past hour and have uploaded photos or tweeted about a specific event*" and "*retrieve all cargo trains that passed yesterday from the surrounding area of Berlin and were transporting agricultural products or were heading to Poland*". Such queries are important for a large number of applications in many domains, including location-based services, fleet management, emergency response, and others, and have remained largely unexplored in existing research.

The results of this work on filtering of trajectories have been published in [120].

1.2.2 Spatial-Temporal-Textual Retrieval of Posts

Analyzing posts made by users on social networks is valuable for a wide range of applications, such as event detection [138, 96], topic detection [35], and opinion mining [156]. Users often want to browse and navigate across content in microblogs to track and monitor the evolution of events and stories as they unfold in the dimensions of space and time. However, this is not trivial for events and topics with a large span in space and time due to the potentially very large number of relevant posts. Thus, our goal in this part is to introduce a novel type of spatial-temporal-textual query that returns a selected set of k results based on the spatio-temporal distribution of the posts in order to facilitate exploratory search, and to devise algorithms for efficiently evaluating the query. To this end, we propose the concepts of *spatio-temporal coverage*, which favors posts from dense regions, and *spatio-temporal diversity*, which ensures that results are well-dispersed over the query region. The unison of these two criteria allows us to identify a small diverse set of representative posts for the query, which makes this method suitable for exploratory analysis of a large number of relevant posts.

Our method for top-k retrieval of posts has appeared in [122].

1.2.3 Continuous Summarization of Streams of Posts

As discussed earlier, examining user posts on social networks is invaluable for several tasks, such as monitoring local events and topics, and understanding public opinions and sentiments. However, the continuous generation of large volumes of geotagged posts makes it difficult and even, impractical to keep track of the entire data stream over time as new messages are posted. Due to the overwhelming amount of information and the inherent repetition and redundancy in this user-generated data, it is often sufficient or desirable to present a concise summary of the evolving stream, which is kept up-to-date as fresh posts arrive. Therefore, our goal in this part of our work is continuous spatio-textual summarization, i.e., maintaining a diverse collection of relatively few, representative posts over the stream. To restrict the summarization to the recent posts only, a time-based sliding window is used, and the results are updated dynamically with each window slide. Moreover, to construct the summary and to estimate its quality, we define the criteria of *spatio-textual coverage* and *spatio-textual diversity*. Here, coverage measures the extent to which the summary captures the original information, whereas diversity ensures novelty among the results. We present and evaluate several alternative strategies with the objective of achieving low execution times without sacrificing the quality of the summary.

This work on continuous spatio-textual summarization of streams has been submitted for publication [141].

1.2.4 Discovery and Exploration of Locally Trending Topics

People use social networks to post information about their surroundings, activities, and opinions. As a result, analysis of geotagged posts can provide important real-time insights into local views and trends. Thus, in this part of our work, we investigate the use of social networks for discovering and exploring currently trending topics. Since the subjects being discussed on social networks tend to vary from region to region, we partition space into smaller regions and identify local topics by aggregating geotagged posts lying within a region. To find topics that are currently popular, a sliding temporal window is used to limit messages and identify topics in a streaming fashion. Moreover, it is often important not only to find popular topics and events, but also to find a small subset of messages that can be used to provide an overview of the topic and to facilitate further exploration. This is necessary because each topic might have thousands of messages associated with it, and thus it is not straightforward for a user to get a quick grasp of the topic's context. Therefore, in this part, our goal is to develop a system for the detection and summarization of locally trending topics in microblog posts.

Our system for topic discovery and exploration has appeared in [124].

1.2.5 Mining Associated Location Sets

By uploading photos, posting tweets, or checking in at various locations, users moving around a city tend to generate *digital trails* of their activities. These trails enable the analysis and extraction of groups of associated locations based on the activities of city-dwellers or visitors. In turn, the discovered associations between locations can be used to build smarter location-based services and better understand how people experience their urban environment. In this part of our work, given a set of keywords, our goal is to find groups of locations that are associated with each other and with the given keywords via user trails. The intuition is that locations that tend to lie together on user trails (i.e., are popular together) and be associated with a similar set of keywords (i.e., are collectively relevant to the query) are likelier to hold a latent thematic connection.

This work on mining associated location sets has appeared in [121] and [125].

1.3 Thesis Outline

Having outlined the motivations and objectives of the problems we investigate, we now proceed to briefly explain the structure of the remainder of this thesis. The next chapter systematically surveys related work on spatial keyword queries by presenting a list of criteria and grouping existing works into categories based on these. The first section of the chapter is devoted to the classic and potentially most prevalent type of spatial keyword queries, called standard queries, due to the extensive prior research on them. Following this, we move on to other categories of related work by going through the criteria progressively. The subsequent five chapters explain the problems dealt with in this thesis. Chapter 3 examines the problem of retrieving movement trajectories matching a spatial-temporal-textual filter and proposes two hybrid indexes for query processing. The kCD-STK query for finding the top-k posts for a spatial-temporal-textual filter is presented in Chapter 4, along with baseline and index-aware algorithms for evaluating it. In the following chapter (Chapter 5), algorithms for computing diversified spatio-textual summaries of streams of posts are presented. The different methods are also extended to take advantage of spatio-textual grouping of posts and are compared on grounds of quality of the summary and performance of the computation. Chapter 6 describes the architecture and demonstration of μTOP, a system for detection and exploration of locally trending topics in microblogging platforms. The task of finding associated sets of locations based on user mobility and behavior is analyzed in Chapter 7, where baseline and optimized approaches based on the Apriori algorithm [2] are explained for the problem. Lastly, Chapter 8 concludes this thesis by presenting a summary of our contributions and identifying potential avenues for future research.

CHAPTER 2

LITERATURE SURVEY

There has been extensive research on spatial keyword queries in the recent years and a variety of query types and query processing techniques have been proposed so far. Due to the large body of work on this subject, in this chapter, we attempt to group together related approaches according to several criteria in order to review them more systematically. Specifically, we devote the first section (Section 2.1) to the fundamental type of spatial keyword queries, termed *standard queries* [22, 39, 42], and categorize the proposed techniques in this area following the approach in [32] based on the types of indexes used and the way these indexes are combined. Subsequently, we review other approaches by going through our proposed list of criteria for classifying the existing literature, namely the *granularity of results*, the *significance of co-location in relevance estimation*, the *strategy for query evaluation*, the *relevance of additional object attributes*, the *type of object geometry*, and the *underlying space*, and discuss the relevant works for each. An overview of our categorization scheme is presented in Table 2.2. Finally, Section 2.9 summarizes the conclusions of this chapter.

2.1 Standard Queries

Standard spatial keyword queries involve ad hoc searches for POIs over typically static objects that return either all or top-k relevant objects. The query contains a spatial component and a textual component. The textual part comprises a set of keywords, which can be used either for ranked retrieval, e.g., ranking documents or web pages based on term frequencies, or as boolean filters, e.g., when searching through short text messages or metadata matching one or more keywords. Similarly, the spatial part may specify a location, in which case the results can be ranked by proximity

Table 2.1 Indexes for standard spatial keyword queries (extended from [32]).

Index	Spatial part	Textual part	Coupling	BRQ	BkQ	TkQ
ST [155]	Grid	Inverted File	Spatial-first	✓		
TS [155]	Grid	Inverted File	Text-first	✓		
IF-R*-Tree [183]	R*-Tree	Inverted File	Text-first	✓	△	
R*-Tree-IF [183]	R*-Tree	Inverted File	Spatial-first	✓		△
SF2I [36]	SFC	Inverted File	Spatial-first	✓		
KR*-Tree [83]	R*-Tree	Inverted File	Tightly coupled	✓	△	
IR^2-Tree [46]	R-Tree	Bitmaps	Tightly coupled	△	✓	
IR-Tree [40, 163]	R-Tree	Inverted File	Tightly coupled	△	△	✓
IR-Tree [107]	R-Tree	Inverted File	Tightly coupled			✓
SKIF [97]	Grid	Inverted File	Tightly coupled	✓		
SKI [26]	R-Tree	Bitmaps	Spatial-first		✓	
S2I [140]	R-Tree	Inverted File	Text-First	△	△	✓
WIBR-Tree [164]	R-Tree	Inverted Bitmaps	Tightly Coupled	△	✓	
SFC-QUAD [38]	SFC	Inverted File	Tightly Coupled	✓		
IL-Quadtree [173]	Quadtree	Inverted File	Tightly Coupled	△	✓	△
I^3 [175]	Quadtree	Inverted File	Tightly Coupled	△	△	✓
RCA [176]	SFC	Inverted File	Text-First	△	△	✓

to it, or a spatial region, which can act as a boolean filter to retrieve all objects contained inside it. The indexed data usually consists of geotagged descriptions of POIs from different sources, such as Wikipedia, OpenStreetMap, online business directories (e.g., Google My Business[1]), and location-based social networks (e.g., Foursquare). For evaluating the query, most approaches focus on combining textual indexes with spatial indexes to produce hybrid spatio-textual indexes that can prune the search space on both spatial and textual dimensions. A survey and comparison of twelve indexes for standard queries was conducted by Chen *et al.* [32]. The authors categorize the existing works based on the types of indexes used for the spatial and textual components, and the technique for combining the indexes into a hybrid index. This is shown in Table 2.1, where the classification presented in [32] is used and extended by us to include more recent works. Each of the three columns at the end represent a type of standard query (see Section 2.1.1). The ✓mark under a column for a query type signifies that the index is originally developed for this query, whereas the △ symbol means that the index can be easily employed to evaluate this query with zero or minor modifications. Below we explain these queries in more detail.

[1]https://www.google.com/business/

2.1.1 Types of Standard Queries

Based on whether the spatial and the textual parts of the query are used for boolean matching or for ranked retrieval, the following major types of standard queries have been studied in existing literature [32]:

- The **Boolean Range Query** (BRQ) applies a set of keywords and a spatial region as boolean filters, returning all documents contained inside the region and matching the keywords.
- The **Boolean *k*NN Query** (B*k*Q) comprises a set of keywords and a point geolocation. It uses the keywords as a boolean filter and ranks the results based on their proximity to the query location, returning the k nearest neighbors.
- The **Top-*k* *k*NN Query** (T*k*Q) retrieves the top-k documents based on an aggregate score combining both textual relevance to the query terms and spatial proximity to the query location. Here, both the spatial and textual components of the query are used jointly for ranked retrieval. Typically, the spatio-textual score $\phi(o, q)$ of an object o with respect to a query q is defined as a weighted linear combination of its spatial proximity and textual relevance to the query, i.e.,

$$\phi(o,q) = (1-\alpha)\cdot(1-\phi_d(o.l, q.l)) + \alpha\cdot\phi_t(o.\Psi, q.\Psi), \tag{2.1}$$

  where $o.l$, $q.l$ and $o.\Psi$, $q.\Psi$ are the locations and the keywords of the object and the query, respectively, and $\alpha \in [0, 1]$ is the weighting factor. $\phi_d(\cdot,\cdot)$ denotes the spatial distance, e.g., Euclidean distance, whereas $\phi_t(\cdot,\cdot)$ signifies the textual relevance, e.g., cosine similarity or tf–idf weighting.

To illustrate, searches, such as *"Find all five-star hotels in Berlin city center"* and *"Find restaurants near me that serve Schnitzel and Apple Strudel"*, are examples of standard queries. Here, since the former requests all the hotels matching the term *"five-star"* and located within *"Berlin city center"*, it highlights a BRQ. On the other hand, the latter can represent either a B*k*Q or a T*k*Q based on whether the terms *"Schnitzel"* and *"Apple Strudel"* are used for retrieving the restaurants matching these or for ranking the results in combination with spatial proximity to the user's current location, respectively.

In addition to the above classification, queries can be either **conjunctive** (i.e., following AND semantics) or **disjunctive** (i.e., following OR semantics) depending on whether objects matching *all* or *at least one* of the query keywords are requested, respectively.

2.1.2 Types of Indexes

Essentially, the indexes proposed by the different approaches for processing standard queries are hybrid structures comprising a spatial indexing part and a textual indexing part. Thus, the choice of indexes used to form these hybrid structures is an important differentiating factor. In existing work, there are mainly three types of spatial indexes that have been used: (1) *tree structures*, such as R-trees [79], quadtrees [64], or kd trees [13], (2) *space filling curves*, e.g., the Z-order curve [128] and the Hilbert curve [85], and (3) *grids* [133].

Among the tree structures, R-trees index the data itself, while quadtrees index the data space. R-trees and kd trees are suitable for fine-grained indexing through minimum bounding boxes, but since these boxes tend to overlap, multiple sub-trees have to be traversed. For point data, it might be more efficient to use a coarse-grained index, e.g., a quadtree. However, quadtrees become less suitable for storing polygons because in case an entry overlaps multiple leaf nodes, it has to be duplicated across all those nodes [68]. On the other hand, space filling curves provide a linear ordering of documents based on their locations, where documents close to each other in space tend to lie close to each other on the curve. This property can be used to store the documents as a sorted list and to retrieve those lying close to the query location through sequential access from the query's position on the list. However, this technique also produces false positives that need to be filtered out.

The textual index can be either an *inverted file* or a *signature file*. A simple inverted file consists of a list of terms in the corpus, called a *vocabulary*, and for each term, an *inverted list*, i.e., a list of identifiers of documents containing the keyword. On the other hand, a signature file uses bits to mark the presence of terms through hashing [184]. A bitmap is a kind of signature where each term is allocated a separate bit in the signature. Thus, several indexes have been proposed employing different combinations of these structures.

2.1.3 Index Combination Technique

Another important difference between the different approaches is how loosely or tightly the spatial and textual structures are combined. In case of loosely coupled structures, the indexed objects are filtered sequentially by the spatial and textual parts. Thus, depending on which dimension is used first while partitioning the dataset, the combination follows either a *text-first* or a *spatial-first* approach [38]. In the first case, for example, the top-level index can be an inverted file, in which the postings in each inverted list are indexed by an R-tree. While processing the query, the objects are first filtered by the inverted file, and then the resulting candidates are checked against the R-tree. Instead, in the second case, the top-level index can be an R-tree, with inverted files attached to each leaf node. Characteristic examples of the two combination schemes are the IF-R*-tree and the R*-tree-IF [183]. Evidently, these loosely coupled approaches are not very efficient as the number of false positives produced after the first filtering step can be quite high.

Tightly coupled hybrid indexes overcome the limitation of their loosely coupled counterparts by integrating textual information into spatial indexes and vice versa. Thus, during query processing, the search space can be pruned using both spatial and textual criteria. For example, another index structure that is based on the R-tree and inverted files, but combines them more tightly, is the IR-tree [40, 163]. The IR-tree improves upon the R*-Tree-IF structure and can be used for both boolean and top-k retrieval of geotagged documents. In order to prune the tree search, it augments the nodes of the R-tree with a *pseudo document* that contains the distinct keywords inside the documents in the sub-tree rooted at the node. Moreover, for each term in the pseudo document, it also stores the highest textual score in the node's sub-tree. The pseudo document can therefore be used to compute the highest textual relevance of any object in the node's sub-tree. Combining this with the `mindist` value of the node's Minimum Bounding Rectangle (MBR) generates an upper bound for the spatio-textual score of objects under the given node, based on the weighted sum definition of score (Equation 2.1). As a result, during query processing, the tree nodes can be ranked by their upper bound scores. This allows a best-first traversal [86] of the tree and early termination by pruning non-promising regions while finding the top-k objects.

Among the works evaluated in [32], for the boolean range query, the SFC-QUAD index [38] outperforms others in terms of disk space usage and runtime performance [32], and thus has also been used in our work. The structure of SFC-QUAD is depicted in Figure 2.1. It consists of an inverted index over the keywords in the entire dataset, where the inverted lists are compressed using a block compression algorithm [167] before being stored on disk. For spatial indexing, a quadtree whose nodes are ordered using the Z-curve is used. To integrate spatial information into the inverted file, each document is assigned an identifier based on its position on the curve and the documents in the inverted lists are arranged in the order of the identifier. Thus, it utilizes the nature of the Z-curve to ensure that documents in the query region also lie close to each other on the list in order to reduce the number of disk I/Os. During

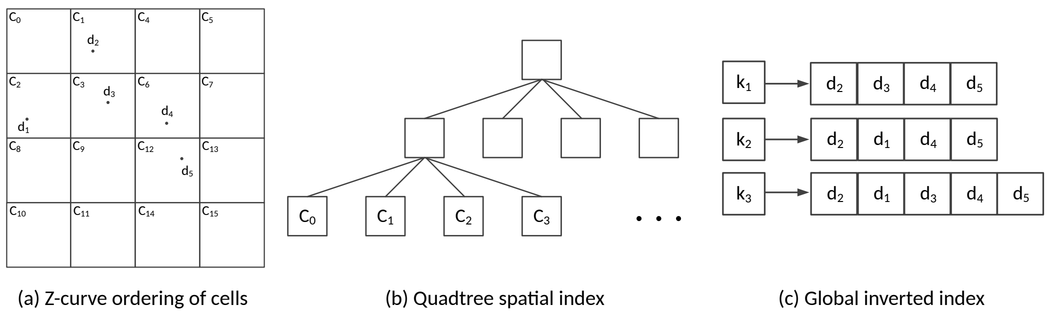
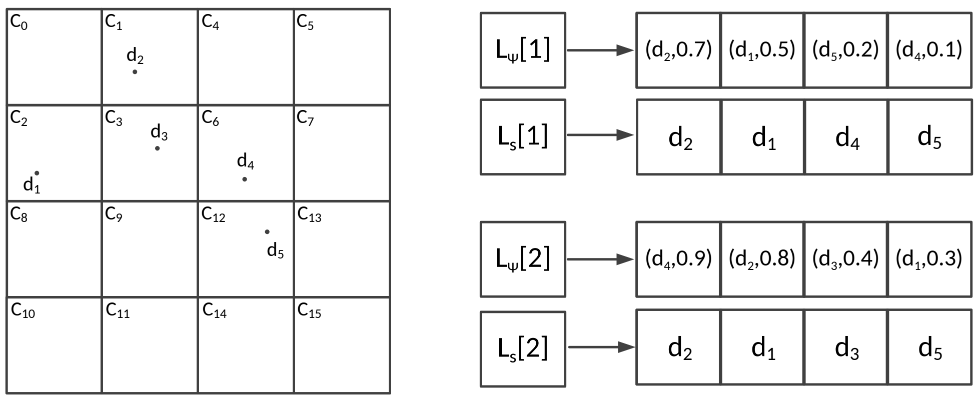


Fig. 2.1 Components of the SFC-QUAD index.

query evaluation, the quadtree is used to find a small number of ranges of document identifiers that are subsequently read from the inverted index to find documents containing all query keywords. The final refinement step eliminates the false positives, i.e., documents outside the query region.

Similarly, for the Top-*k* *k*NN query, two state-of-the-art approaches (proposed after the survey in [32]), which are also used in our work, are the I^3 hybrid index [175] and the RCA algorithm [176]. The I^3 index maintains a quadtree for each keyword, indexing the documents containing it. Each keyword is used as a key in a lookup table and is associated with a pointer. If the documents containing this keyword can fit in a single disk page, the pointer links directly to that page; otherwise, it points to the root of a quadtree which spatially indexes the relevant documents. The leaf nodes of the quadtree point to the disk pages where the documents are stored. Given this index, a spatio-textual query is processed as follows. First, the relevant keywords are identified, depending on whether OR or AND semantics are used. Then, the relevant documents are searched accordingly, depending on whether the keyword is *dense* or not, i.e., if the number of objects containing the keyword exceed the capacity of a disk page or not. For keywords that are not dense, the relevant documents can be retrieved by a single page access. For dense keywords, the nodes of the quadtree are traversed, checking whether the spatial extent of a node intersects with the spatial bounding box of the query.

The RCA approach uses only an inverted index and is depicted in Figure 2.2. In particular, it maintains two inverted lists for each keyword. The first is a standard inverted list (shown as L_ψ in the figure) that stores the documents containing the keyword in decreasing order of relevance. The second one (L_s in the figure) contains documents according to the Z-order encoding of their coordinates. Query processing exploits the following property of the Z-order encoding. Assume a spatial bounding box R, with z_{min} and z_{max} being the Z-order encodings of its top-left and bottom-right



C_0 C_1 C_4 C_5
C_2 C_3 C_6 C_7
C_8 C_9 C_{12} C_{13}
C_{10} C_{11} C_{14} C_{15}
d_1 d_2 d_3 d_4 d_5
$L_\psi[1]$ $(d_2,0.7)$ $(d_1,0.5)$ $(d_5,0.2)$ $(d_4,0.1)$
$L_s[1]$ d_2 d_1 d_4 d_5
$L_\psi[2]$ $(d_4,0.9)$ $(d_2,0.8)$ $(d_3,0.4)$ $(d_1,0.3)$
$L_s[2]$ d_2 d_1 d_3 d_5

(a) Z-curve ordering of cells (b) Inverted index

Fig. 2.2 Components of the RCA approach.

corners, respectively. Then, the Z-order encoding of any point that lies within R has a value $z \in [z_{min}, z_{max}]$. This allows to efficiently process top-k queries using an adaptation of the CA algorithm [59] for rank aggregation. This has the advantage that the method can more easily be implemented and deployed in existing search engines, since they already rely on inverted indexes for document search.

This section presented an overview of relevant works on standard queries. Hereafter, we discuss other types of spatial keyword queries by going through a set of classification criteria, beginning with approaches that identify areas or groups containing multiple objects that collectively satisfy a given query.

2.2 Granularity of Results

The approaches described so far focus on the retrieval of *single* objects that users might be interested in. However, very often the nature of the query calls for decreasing the result granularity and presenting *sets* of POIs, instead of individual POIs, where the objects in each set collectively match user requirements. Below we discuss different lines of work in this direction, focusing on retrieving different types of result sets, such as *areas of interest* or *collections of spatio-textual objects*.

2.2.1 Areas of Interest

Due to the large quantity of available spatio-textual data, on many occasions, it is desirable to return entire areas that contain several relevant POIs for user exploration, instead of the POIs themselves. Skoutas *et al.* [146] focus on streets as the desirable

Table 2.2 Categorization of existing work on spatial keyword queries.

Query		Object Geometry	Result Granularity		Co-location Aware	Execution Strategy		Underlying Space	Additional Attributes
Type	Works		Level	Cardinality		Continuous	Parallel		
Standard	Table 2.1	Point	Object	Single				Euclidean	
	[36, 60]	Region	Object	Single				Euclidean	
	[139, 172]	Point	Object	Single				Road Network	
	[115]	Point	Object	Single			✓	Road Network	
Areas of Interest	[146, 23]	Point	Region	Single	✓			Road Network	
	[61, 62]	Point	Region	Single	✓			Euclidean	
Collective	[174, 21, 77, 24]	Point	Object	Collection	✓			Euclidean	
	[70]	Point	Object	Collection	✓			Road Network	
Preference-Aware	[154]	Point	Object	Single	✓			Euclidean	
	[49]	Point	Object	Single	✓		✓	Euclidean	
Prestige-Based	[20]	Point	Object	Single	✓			Euclidean	
Moving	[89, 165, 76]	Point	Object	Single		✓		Euclidean	Time
	[75]	Point	Object	Single		✓		Road Network	Time
Publish/Subscribe	[31, 34, 88, 161, 33]	Point	Object	Single		✓		Euclidean	Time
	[104]	Region	Object	Single		✓		Euclidean	Time
	[37, 160]	Point	Object	Single		✓	✓	Euclidean	Time
Big Data Systems	[117]	Point	Object	Single		✓	✓	Euclidean	Time
	[114]	Point	Object	Single			✓	Euclidean	
Posts	[130, 87]	Point	Object	Single				Euclidean	Time
Trajectories	[41]	Point Sequence	Object	Single				Euclidean	Time
Geo-Social	[4]	Point	Object	Single				Euclidean	Connectivity
	[94]	Point Collection	Object	Single			✓	Euclidean	Connectivity
Join	[16]	Point	Object	Single				Euclidean	
	[109, 110]	Region	Object	Single				Euclidean	
	[9, 136]	Point	Object	Single			✓	Euclidean	
	[54]	Point Collection	Object	Single				Euclidean	
Reverse	[112]	Point	Object	Single				Euclidean	
	[69]	Point	Object	Single				Road Network	
Direction-Aware	[103]	Point	Object	Single				Euclidean	

unit of user interest and investigate the problem of finding *Streets of Interest* (SOIs) for a given set of terms. They define an SOI as a collection of road network segments containing one or more POIs within a distance threshold from it whose description matches the query keywords. Given this definition, the candidate streets are ranked based on the density of POIs within the distance threshold and the top results are returned. For query processing, a combination of a spatial grid index and an inverted index is used. Moreover, the work also deals with visually describing SOIs using geotagged photos. For this, a set of k geotagged photos (e.g., from Flickr) is identified for a given SOI whose descriptions are not only spatio-textually relevant, but also spatio-textually diverse, in order to provide a quick visual overview of the street.

Another line of related work deals with finding *Regions of Interest* (ROIs) [23, 61, 62]. Given a size constraint, the goal here is to find regions where the POIs inside are relevant to the query keywords and collectively maximize an objective score, such as textual relevance, popularity, or diversity, while satisfying the size constraint.

For this, Cao *et al.* propose the *Length Constrained Maximum-Sum Region* (LCMSR) query, where the size constraint is specified in terms of road network length and the objective score is the sum of the scores of the objects inside the region [23]. Thus, the regions found can be of any shape, depending on the road network topology. [61] targets a similar problem, called *Best Region Search* (BRS), of finding rectangular regions of specific dimensions for the general case where the objective is a monotone submodular function. Due to this, the query can be used to find regions with the most diverse set of POIs or with the highest influence among users by using monotone submodular functions to represent diversity or influence, respectively. Based on the work in [61], a system for region search and exploration is presented in [62].

2.2.2 Object Collections

Collective Spatial Keyword (CSK) queries [174, 21, 77, 24, 70] extend standard queries with the goal of satisfying complex information needs. For example, a tourist arriving in a city might want to have coffee, see the river, and visit a museum, and thus might look for places using the terms *'coffee, river, museum'*. Evidently, such a search could be better satisfied *collectively* by a group of objects, rather than *individually* by single objects. The *m Closest Keywords (mCK) query* introduced in [174] was the first work to address this challenge. Given a database of spatio-textual objects and m keywords, this query retrieves a set of objects that (1) together contain all the m keywords in their keyword sets, and (2) are located as close to each other as possible. For query processing, an augmented R*-tree [12] structure, called bR*-tree, is proposed, which stores a bitmap at each node summarizing the keywords in the sub-tree. Additionally, for each keyword in the sub-tree, an MBR is maintained, which represents the spatial extent of the objects containing the keyword. Further in this direction, Cao *et al.* show that the mCK query is NP-hard and devise approximation algorithms for faster computation [24].

A similar variant, called the *Spatial Group Keyword query*, is defined in [21], where, in addition to a keyword set, a query location is also supplied. Here, the retrieved objects need to be as close to the query location as possible, and optionally in proximity to each other. Both instances of the problem are proven to be NP-complete; thus, both exact and approximate solutions are presented [21]. Similarly, [70] addresses the problem of efficient evaluation of CSK queries for objects located on a road network, instead of the Euclidean space.

We revisit the problem of finding collections of locations for a given set of keywords in Chapter 7. However, in contrast to the aforementioned works, there our goal is to find collections of locations that are associated based on user trails derived from geotagged posts. Thus, while the existing works mainly focus on optimizing for spatial proximity and ignore user behavior, we focus on retrieving sets of associated locations leveraging user mobility and behavior as the evidence and measure of strength of the association. Consequently, our work is able to capture latent thematic associations between locations that might be overlooked by similar works.

2.3 Co-location Awareness

A significant limitation of the majority of the approaches for spatial keyword query processing is that they define the relevance of an object to a query as a sole function of the attributes of the object itself, thus assuming that it is independent of other objects in the dataset. However, this is seldom the case in real-world scenarios, where very often the appeal of a location is also influenced by other POIs in its vicinity. Below we present the approaches that consider the significance of co-location of objects while ranking the results.

2.3.1 Preference

The *Top-k Spatio-Textual Preference Query* [154] retrieves objects based on the quality of other facilities in their vicinity. Here, the objects being retrieved are called *data objects* (e.g., hotels) and the facilities in the neighborhood are called *feature objects* (e.g., restaurants). Each data object has a location, whereas each feature object additionally contains a non-spatial score, such as a rating, and a textual description. Thus, an example of this query would be *"Find hotels that have a highly rated Italian restaurant in the vicinity serving espresso"*. In [154], first, a baseline approach is proposed, which computes the score of all data objects, and then reports the k data objects with the highest score. Next, an improved approach, which scans promising feature objects first and then finds data objects in their vicinity, is presented. To identify relevant and highly ranked feature objects, the authors propose to modify a spatial index, such as an R-tree, to build a four-dimensional index, called the SRT-index, on the spatial coordinates, the non-spatial score, and a value for the keywords based on the Hilbert curve. A variant of this problem is discussed in [49], where the data is distributed on multiple processing nodes and query processing is carried out in parallel using the MapReduce [47] based Hadoop framework[2].

[2]http://hadoop.apache.org/

2.3.2 Prestige

[20] defines the concept of *prestige-based relevance* to rank those results higher that are not only close to the query, but also have other objects nearby that are relevant. For evaluating the query, the authors build a graph on the objects by connecting those that are sufficiently close in space and similar in textual descriptions. Then, they use a technique similar to PageRank [74] to assign prestige values to the nodes. A distinguishing feature of this approach is that despite the fact that a place does not match any query keyword, it might still be returned as a result due to the effect of neighboring places that are relevant.

Co-location of POIs is also important for the retrieval of areas of interest and object collections (Section 2.2), where regions and groups comprising multiple POIs are returned, respectively, based on the attractiveness and proximity of POIs within them.

Similarly, in our work, we propose the concepts of *spatio-temporal coverage* (Chapter 4) and *spatio-textual coverage* (Chapter 5), which allow us to define the relevance of a post indirectly in terms of its similarity with the other objects in the dataset. By combining coverage with diversity of the result set, we are able to derive a measure of its representativeness, and thus its suitability for exploratory analysis and summarization.

2.4 Query Execution

Until now, we have discussed several different classes of spatial keyword queries targeting different use cases. Nonetheless, these efforts mainly focus on scenarios where both the query and the objects are static, and where queries are evaluated only once in an ad hoc fashion. On the other hand, the main focus of this thesis is on the analysis of geotagged posts, which are dynamic objects being produced continuously in large volumes and at high rates. This calls for continuous query execution and monitoring of results over time, which we discuss in this section. Furthermore, another important area of research for analyzing data at a large scale are techniques based on parallel and distributed architectures. Although this thesis does not focus on the development of parallel and distributed solutions for query processing, this is potentially a very promising area for future research, and hence is also discussed here.

2.4.1 Continuous Evaluation

In contrast to ad hoc or snapshot variants, continuous queries generally specify a filter over an incoming stream of data, where results are updated as new objects in the stream arrive and expire. Here, there are two major bodies of relevant research, namely the moving spatial keyword query and spatio-textual publish/subscribe systems.

Moving Query

The *moving top-k spatial keyword query* maintains the top-*k* relevant spatio-textual objects for a moving user in real-time [89, 165, 75]. The usual approach is to define the concept of a *safe region* within which a result set is valid [89, 165]. If and when the query object exits the safe region, the result set needs to be updated. This method, however, is only suitable for objects moving in the Euclidean space. In [75], both the user and the objects are confined to a road network, and techniques for query processing through incremental expansion of the network from the query position as well as the relevant objects are presented.

[76] proposes a publish/subscribe system that continuously monitors moving users subscribing to dynamic location-aware events (e.g., social network messages). The subscriptions are modeled as boolean expressions along with a notification radius. To reduce communication overhead, the authors exploit the idea of safe regions and propose the concept of *impact regions* for subscribers to determine whether their safe regions can be affected by newly arriving messages. Furthermore, to support matching past published events to subscribers, the authors propose a Boolean Expression Quad-Tree (BEQ-Tree) structure for indexing events to reduce response time.

Publish/Subscribe

The problem of continuously maintaining the most relevant results over a stream of spatio-textual documents from different sources, such as social networks, has been investigated in recent works on spatio-textual publish/subscribe systems. [31] proposes the Inverted File Quad-tree (IQ-tree) for indexing a large number of subscriptions in order to efficiently identify queries for which an incoming object might be a candidate. The IQ-Tree is essentially a quadtree augmented with inverted indexes at its nodes. The subscription is used as a boolean filter to continuously return *all* objects lying inside the query region and time window, and matching query keywords. The work in

[104] also focuses on the problem of delivering textually matching (i.e., containing all query keywords) and spatially relevant (i.e., overlapping query MBR) messages to subscriptions. To index the large number of subscriptions, the authors propose to use an R-tree augmented with textual descriptions of subscriptions at its nodes to store the spatial and textual attributes of the subscriptions. In the same spirit, the parameterized technique in [88] weighs textual relevance and spatial proximity in a combined spatio-textual similarity measure and proposes a filter-and-verification framework to deliver all messages for a subscription with similarity higher than a given threshold. In particular, the authors introduce three alternative filtering schemes: a *spatial-oriented prefix* based on inverted indexes that capitalizes on maximum spatial similarity, a *region-aware prefix* based on hierarchical spatial indexes (e.g., R-trees) so that subscriptions can be grouped by locality, and a *spatio-textual prefix* utilizing multiple keywords for pruning. A cost model is suggested so that the best filtering strategy can be selected.

On the other hand, [34] combines the criteria of textual relevance, spatial proximity, and a temporal decay-based recency function to find the top-k results for a query over a stream of spatio-textual objects. A prototype based on this approach and the approach proposed in [31] for continuously processing boolean and top-k queries is presented in [33]. Similarly, Wang *et al.* [161] concentrate on the same problem as in [34], but using a sliding window instead of a recency function, to find the k most relevant messages. As in [31], they also use a subscription index that is a combination of a quadtree with inverted files at the leaf nodes. Moreover, for maintaining the top-k results over the sliding window, they employ a cost-based k-skyband, which is an extension of the k-skyband proposed in [129].

Despite the fact that continuous queries for spatio-textual data have received significant attention recently, none of these works investigate the problem of summarization of spatio-textual streams via diversification, which is the focus of Chapter 5. The closest to our approach is [30], where the authors analyze the problem of *diversity-aware top-k subscription queries* over textual streams. Qualifying documents are ranked by textual relevance, temporal recency, and result diversity according to respective score functions. However, in contrast to our work, to process incoming documents efficiently, the proposed method employs a rather restrictive process. Each new document is compared only to the oldest one in the current result set, and if it improves the objective score, the replacement is made, otherwise the document is discarded. Moreover, the considered documents do not have a spatial attribute.

2.4.2 Parallel Processing

The growing amount of spatio-textual data also poses the challenge of scaling query processing to clusters of computers [116]. The idea behind these approaches is to divide the original task into subtasks for execution on different machines. Each cluster node has its own memory, and communication between nodes mainly takes place through message passing. The works on this subject can be grouped into two broad classes: (1) systems for processing large-scale spatio-textual data, and (2) adaptations of existing spatial keyword queries and algorithms for distributed settings.

Big Data Systems

MapReduce-based frameworks, such as Hadoop[2] and Spark[3], allow the development of distributed solutions using elementary programming operations. However, as these are general purpose frameworks, they lack optimizations and support for spatial or spatio-textual data. [162] addresses this challenge by detailing an approach for indexing spatial data stored in the Hadoop Distributed File System (HDFS). The authors propose a two-tier index comprising a single global index and multiple local indexes. The global index is used to distribute the data across the nodes, whereas the local index is constructed on the data at each processing node. SpatialHadoop, an extension of Hadoop with native support for spatial data is described in [55] and [56], and a comparison of different spatial partitioning techniques supported by it is presented in [57]. Similarly, GeoSpark [170, 171] extends Spark for spatial query processing and analytics. A survey of approaches in the area of processing large-scale spatial data can be found in [58] and [81]. LocationSpark [151] goes a step further in this direction by also supporting spatio-textual analytics, in addition to spatial queries and analytics, over Spark. Support for spatial data has also been integrated into distributed databases. A prominent example is GeoMesa [65], which integrates spatio-temporal indexing into non-relational databases, such as Accumulo[4]. Recently, a distributed system extending the Storm[5] stream processing framework, called Tornado, for executing continuous spatial keyword queries over data streams has been introduced in [117]. It uses a distributed spatio-textual index to ensure that the data necessary for a specific query resides on the same node. Furthermore, the index also adapts to changes in data distribution and query workload by re-distributing the processing across nodes. Another example in this domain is sksOpen, which allows

[3]http://spark.apache.org/
[4]http://accumulo.apache.org/
[5]http://storm.apache.org/

visualization and querying of large-scale spatio-textual data [114, 177]. The system supports the boolean *k*NN query by employing a variant of the indexing mechanism proposed in [26], which uses a combination of an R-tree and inverted files containing bitmaps for each term.

Specific Algorithm Implementations

In addition to these new systems, several existing methods have been extended to work in a distributed context [9, 115, 49, 37, 160]. [115] develops distributed techniques to handle the boolean range query and other query variants on road networks that use a set of keywords, a distance threshold, and a location as query inputs. To evaluate the queries, a distributed index is created that allows each node to carry out its computation independently, and thus minimize communication overhead. Similarly, [37] presents a distributed solution for dealing with a large volume of subscriptions arriving frequently in publish/subscribe systems. The authors describe a technique for partitioning the workload of insertions and deletions of queries, and the matching operations between queries and objects over a cluster of servers based on both the spatial and textual distributions of the data. Moreover, approaches for adjusting computation load dynamically are also presented and evaluated. [160] extends the work in [161] to build a distributed publish/subscribe system on top of Storm for supporting higher throughput and scalability. Different mechanisms for distributing the subscriptions and messages are examined and compared to find one that minimizes the communication cost and achieves the best performance.

2.5 Relevance of Additional Attributes

In addition to spatial and textual information, most sources of spatio-textual data offer other kinds of information in the form of attributes and metadata that can be exploited to enable a rich variety of analyses and features. We discuss some of the works in this direction below.

2.5.1 Temporal Data

Spatial keyword queries are typically oblivious of the temporal information associated with objects. However, driven by the growing amount of Web data containing timestamps and spatial footprints, the need for combining keyword search with spatial and temporal filtering has been recognized. In the following, we outline research in the area of spatial-temporal-textual query processing.

For querying spatio-temporal posts, such as geotagged tweets, in [130], the authors propose an index that is based on a shallow R-tree, combined with an inverted index at each leaf node to index the terms of the contained documents. In addition, to deal with the temporal dimension, the original document identifiers are replaced with new ones that are assigned to documents chronologically, thus facilitating the retrieval of documents within a given temporal range. Similarly, [87] proposes a disk-based variant of the kd tree for supporting range and top-k queries combining the spatial, temporal, and textual dimensions. Keywords are transformed to numerical values to allow access via the index.

In a related direction, keyword search on trajectories has been studied in [41]. Each trajectory consists of a sequence of geolocations associated with textual descriptions. In [41], given a location and a set of keywords, the goal is to find the top-k trajectories whose textual descriptions cover the given keywords and which have the minimum distance to the given location. The proposed method is based on a hybrid index, called cell-keyword conscious B$^+$-tree, which enables simultaneous application of both spatial proximity and keyword matching.

Other types of queries with spatial, temporal, and textual filtering include the moving spatial keyword query and the publish/subscribe variants. These have already been covered in Section 2.4.

In summary, the amount of work dealing with temporal information available with spatio-textual data has been limited so far. Moreover, the relevant works in this area focus largely on the retrieval of individual posts by either applying spatio-temporal criteria as boolean filters or by using them to rank posts based on spatial proximity and/or recency. These are very different from the related problems studied in this thesis, including spatial-temporal-textual filtering of trajectories (Chapter 3) and the retrieval of top-k posts for a spatio-temporal range and set of keywords based on spatio-temporal coverage and diversity (Chapter 4).

2.5.2 Social Connectivity

The relationship of an object with others in the dataset is a potential measure of its importance or popularity and can be used for ranking the results relevant to a query. Jiang *et al.* [94] formulate the problem of *top-k local user search* in geotagged Twitter data. Given a location, a distance threshold, and a set of terms, the query finds the top-k users who have posted messages on Twitter relevant to the query keywords within the distance threshold from the query location. They propose the notion of a *tweet thread* to estimate the popularity of tweets and users based on the number of responses (i.e., replies and forwards) a message receives. Based on this, a user score combining spatial proximity, textual relevance, and tweet popularity is computed for ranking local relevant users. A distributed index comprising a quadtree and an inverted index is proposed, and the Hadoop framework is used for implementing index construction and query processing on a cluster of computers.

A framework supporting queries capturing spatial, social, and textual criteria for finding top-k users, POIs, or keywords is proposed in [4]. Three different variants of geo-social keyword search are proposed: (1) given a location and a set of terms, identify the top-k users based on spatial proximity, network popularity, and textual relevance, (2) given a user and a set of terms, find the top-k POIs based on spatial proximity, the number of check-ins by the user's friends, and their textual relevance, and (3) given a spatial range, return the top-k keywords based on their frequency in pairs of friends located within the area. A hybrid structure indexing users and POIs based on all the three attributes, and a ranking function combining their partial scores on each dimension are used for query evaluation.

In our analysis, we do not utilize social information available with the posts (e.g., message replies and forwards, or friendship information), except the identifier of the user who created the post in order to group posts made by the same user together.

2.6 Type of Object Geometry

In this thesis, we deal with geotagged posts and represent the spatial attributes of the posts by point locations. However, it is noteworthy that not all the existing approaches for spatial keyword query processing assume that objects' geolocations are points. In fact, several works dealing with larger object footprints have been proposed [36, 60, 109, 110] as this more closely fits many practical use cases, such as modeling regions of user activity for user profiling in social networks and representing territories of animals for wildlife monitoring [60]. For example, in [36], the authors geocode the documents to extract their spatial footprints in the form of one or more non-contiguous regions, each having a non-negative value associated with it. The query comprises a set of terms and a region, and only considers objects that contain all query terms and a non-empty intersection with the query area. The objects in the result set are ranked according to their textual relevance to the query and the extent of overlap between the objects' and the query footprints. The authors evaluate both spatial-first and text-first methods for query processing, considering them as baselines, and conclude that the spatial-first strategy performs better. An optimized approach is also presented that uses Hilbert Curve ordering of the footprints to organize the documents in an inverted file on disk.

A slightly different problem is analyzed in [60], where the goal is to find all objects that have a spatial and textual similarity to the query higher than specified threshold values. The spatial similarity is measured through the extent of intersection between the query and the object region, and is used to define a *spatial Jaccard similarity* measure, while the textual relevance is computed using the weighted Jaccard similarity between the query terms and the object's keywords. A grid is used to construct the spatial signature of an object, whereas the keywords are used as its textual signature. These signatures are then used to build a filter-and-verification framework for finding the objects that exceed the similarity thresholds.

2.7 Underlying Space

Although most of the prior works, as well as the problems investigated in this thesis, assume that objects are restricted to Euclidean space, it is important to mention that several works dealing with spatial keyword search on road networks have also been proposed [139, 172, 75, 115, 146, 23, 69, 70]. In this case, the spatial proximity between objects reduces to a function of their network distance, instead of Euclidean distance. Rocha-Junior *et al.* [139] present techniques for processing the top-*k* *k*NN query on road networks. They first propose a basic approach combining a spatio-textual index with the framework proposed in [134], and improve it to produce two new approaches. Zhang *et al.* study the same problem with the goal of returning relevant as well as spatially diverse results [172]. The diversity of the result set is defined using max-sum objective function [73], where the network distance to the query location is used as a measure of relevance and the pairwise network distance between objects is used to estimate the diversity. Thus, the task is to retrieve a set of *k* objects lying within a distance threshold from the query and containing the query keywords that maximize the objective score. To achieve this, a signature-based inverted index structure is used to organize objects for pruning the search space based on the spatial and textual constraints, and an incremental technique for finding the top-*k* relevant and diverse objects is presented.

2.8 Other Types of Queries

Various other types of problems have been studied in the recent years, including spatio-textual variants of popular spatial queries, such as spatial join and reverse *k*NN query. We discuss some of these below.

2.8.1 Reverse Query

The *Reverse k Nearest Neighbor query* finds objects whose *k* nearest neighbors include the query point [99], and has received significant attention in recent years due to its application in finding the influence sets of objects in a database. Although this problem is well-investigated in the spatial databases literature, generally spatial proximity is used as the sole measure of influence, while textual similarity is not accounted for. To that end, the *Reverse Spatial Textual k Nearest Neighbor (RSTkNN) query*, which uses both spatial and textual similarity, is proposed in [112]. A hybrid index structure, called the IUR-tree, which is formed by augmenting the internal nodes of an R-tree, is presented. The IUR-tree stores the union as well as the intersection of the keyword sets of the objects lying inside the nodes in order to generate lower and upper bounds on spatio-textual similarity between objects for pruning the search space during query processing. Furthermore, [69] presents a variant of this problem, where road network distance replaces Euclidean distance as a measure of spatial proximity and keywords are used for boolean filtering, instead of ranking by relevance.

2.8.2 Join

The *Spatio-Textual Similarity Join* finds pairs of objects in a database that are both spatially close and textually similar. This problem has applications in various fields ranging from social networks to duplicate elimination [16]. [16] defines the problem as finding all pairs of objects with a spatio-textual similarity higher than a specified threshold. The authors use a combination of a dynamic grid created during query execution and a set similarity join algorithm to speed up query processing. Parallel techniques for spatio-textual join over a MapReduce-based system are presented in [9]. The work deals with a slightly different problem definition, where for each object p_1, only the most similar object p_2 is returned. Moreover, a custom similarity function of the form $sim(p_1, p_2) = sim_t(p_1.\Psi, p_2.\Psi)/1 + d_s(p_1.loc, p_2.loc)$ is employed, where $sim_t(\Psi_1, \Psi_2)$ is the textual similarity and $d_s(loc_1, loc_2)$ is the Euclidean distance between objects. Rao *et al.* [136] compare different possible approaches for partitioning data while computing the join. These include (1) a *local* strategy, where data is indexed using a spatial index and a string similarity join is applied locally inside the spatial partitions, and (2) a *global* approach that uses a global inverted index to organize the objects and orders the inverted lists spatially using an SFC. In addition, implementations obtained by changing the spatial index (grid vs. quadtree), the set similarity join algorithm (All-Pairs [10] vs. PPJ [166]), and the computation technique (single- vs. multi-threaded) are also compared. [109] and [110] also deal with spatio-textual join, but for regions, by representing objects' spatial footprints via their MBRs. The spatial similarity between objects is thus defined using the extent of overlap between their MBRs. Finally, [54] extends the problem presented in [16] to matching collections of spatio-textual point objects, instead of single objects, based on similarity.

2.8.3 Direction-Aware Query

The standard queries return the results based on distance and textual similarity, without taking the orientation of the query into consideration. The *Direction-Aware Spatial Keyword Search* (DESKS) [103], on the other hand, returns k objects closest to the query location that contain all the query keywords and additionally lie within a direction range from the query location. The direction awareness feature can be especially useful for location-based services for moving objects, where generally results retrieved based on the orientation or direction of movement of the query object could be more relevant than other objects. To efficiently process the query, the authors propose to prune the search space by partitioning objects into sub-regions based not only on their location, but also on their direction relative to the query. For each sub-region, the textual content of the objects inside it is organized using inverted lists to achieve keyword-based pruning.

2.9 Summary

In this chapter, we have surveyed published literature on spatial keyword queries and established a broader context for this thesis. We first discussed the typical class of spatial keyword queries, termed standard queries, in Section 2.1. Subsequently, we reviewed related works by providing a range of criteria that can be used to distinguish them and by grouping them based on these. These included result granularity, co-location awareness, query execution strategy, additional attributes used, object geometry, and underlying space. The results of our categorization are summarized in Table 2.2.

Based on our survey, it is evident that the problems investigated in this thesis present a significant advancement over the state-of-the-art in spatial keyword query processing. Whereas most of the existing works deal with static settings, such as POI search, we focus on dynamic spatio-textual objects in the form of geotagged posts. Thus, where prior approaches largely ignore temporal information that might be available with the objects, our methods in Chapter 3 and Chapter 4 take advantage of the temporal data for filtering the results and for ranking the top results based on spatio-temporal criteria, respectively. Similarly, the techniques in Chapters 5 and 6 use the post timestamps to always present the current results using a sliding window.

The use of geotagged posts also poses the challenge of handling large quantities of data being produced at a rapid pace and keeping the results up-to-date as new posts arrive. Here, although there has been research conducted in the area of continuous queries on streaming spatio-textual data, none of these address the challenge of continuous summarization of spatio-textual streams, which we present in Chapter 5. Similarly, the techniques proposed in Chapter 3 and Chapter 6 also facilitate the exploration of large numbers of posts, by presenting users with a smaller set of representative posts and trending topics, respectively.

Finally, another aspect in which our work contributes to research is by aggregating posts and utilizing them for enriching locations with patterns and insights inferred from user behavior. In this regard, we present two different analyses. The first is a system for the discovery and exploration of locally trending topics in social networks, whereas the second is the identification of associated sets of locations based on geotagged posts. As in our work on mining associated location sets in Chapter 7, CSK queries also retrieve a group of locations for a given set of keywords. However, our approach for addressing the problem is fundamentally different. This is because, instead of optimizing for spatial proximity, we seek to maximize the co-occurrence of the locations in user trails derived from geotagged posts. Thus, our approach is able to capture latent thematic connections between groups of locations evidenced by users' behavior, that might be overlooked by related works.

CHAPTER 3

SPATIAL-TEMPORAL-TEXTUAL FILTERING OF TRAJECTORIES

Having presented an overview of existing work, we now present the first problem examined in this thesis, namely the retrieval of trajectories of moving objects using a spatial-temporal-textual filter.

3.1 Overview

As a result of the growing use of GPS and mobile devices, it has become possible to track the movement of various types of objects, ranging from ships, airplanes, and vehicles to animals and people. Consequently, storing, querying, and analyzing such movement data is becoming increasingly interesting and important for many applications [178, 179]. To that end, several problems have been studied in this area, including range and nearest neighbor queries for moving objects (e.g., [67, 78, 145]) or finding movement patterns and groups of objects that move together in space and time (e.g., [106]). Moreover, these studies have been concerned both with querying and monitoring current and future positions of objects (e.g., [152, 91, 84]), as well as with storing and querying historical trajectories (e.g., [71]). The trajectories are usually a polyline approximation of the original trajectory of the object. Typical examples of such queries are "*find all vehicles that are currently within 1 km of the Brandenburg Gate*" or "*find all users who crossed Alexanderplatz yesterday evening*". Such queries are important for a large number of applications in many domains, including location-based services, fleet management, emergency response, and others. To support the efficient evaluation of such types of queries on moving objects, several spatio-temporal indexes have been proposed [131].

However, these approaches focus on the spatial and temporal dimension when indexing and querying objects, thus ignoring other important characteristics, in the form of textual attributes, that can be used for keyword-based search and filtering of the objects. For example, in fleet management applications, a moving object, e.g., a ship, an airplane, a truck, or a train, may transmit messages during its movement that contain certain information, such as its type, its next destination, the type of cargo it is transporting, or any other status information. This can be used for instance to answer queries, such as "*retrieve all cargo trains that passed yesterday from the surrounding area of Berlin and were transporting agricultural products or were heading to Poland*".

Moreover, users in social networks generate large amounts of content that encompasses spatial, temporal, and textual information. Consider, for example, a traveler who uploads geotagged photos on Flickr or posts geotagged tweets while moving around in a city. The result is a digital trail of photos or tweets, each post being characterized by a location, a timestamp, and a set of tags or keywords. One may then want to evaluate queries, such as "*retrieve all users who have been in the city center of Berlin in the past hour and have uploaded photos or tweeted about a specific event*".

Combining spatial queries with keyword search has been the focus of spatial keyword query processing. However, as discussed earlier, spatial keyword queries are focused on spatial objects that are typically static, i.e., POIs, places, or, more generally, documents associated to locations, where the query point is either static or moving. The problem of efficiently evaluating queries on moving objects that encompass all three dimensions, *spatial*, *temporal*, and *textual*, remains largely unexplored [130, 82, 41].

In this chapter, we focus on spatio-temporal keyword queries on moving objects. In particular, we address the problem of efficient evaluation of queries that perform spatial, temporal, and keyword-based filtering on historical movement data of objects that is additionally associated with textual information in the form of keywords, potentially changing at each timestamp and location. Our methods combine and build upon concepts and techniques for spatio-textual and spatio-temporal queries, proposing algorithms for efficient evaluation of queries that include filtering criteria on all the three dimensions. More specifically, our main contributions here are summarized below:

- We address the problem of spatio-temporal keyword (STK) queries on trajectories of moving objects; this allows to: (a) extend spatio-temporal queries to moving objects that are associated with textual information, and (b) extend spatio-textual queries to objects that are moving instead of static.

- We propose the GKR index, a hybrid index structure that extends a trajectory index to incorporate the textual information associated with trajectory segments.
- We propose the IFST index, a hybrid index structure that extends a spatio-textual index to incorporate the temporal dimension, allowing to deal with moving objects.
- We evaluate the performance of the two approaches by conducting a detailed experimental evaluation using three real-world datasets from two diverse types of sources, including yacht movement tracking data and geotagged images from Flickr.

The rest of this chapter is structured as follows. Section 3.2 provides a background of related work on spatio-temporal queries. Section 3.3 formally defines the STK query considered in this chapter. Then, Section 3.4 presents the indexes and algorithms proposed for STK query evaluation. Our experimental evaluation is presented in Section 3.5. Finally, Section 3.6 concludes the chapter.

3.2 Additional Relevant Background

Due to the extensive amount of research on spatio-temporal queries and due to its relevance to our problem, in the following, we present an overview of the approaches that have been proposed in this area.

Several works focusing on efficient indexing and querying of moving objects have been published so far. A comprehensive survey of spatio-temporal access methods can be found in [127, 131]. Existing indexes are categorized according to whether they index past, current, or future positions of moving objects (or combine all three).

In this chapter, we focus on the first category, namely indexing the past positions of moving objects. One of the main approaches in this category is **SETI** [28]. SETI employs a two-level index structure to handle the spatial and the temporal dimensions. The spatial dimension is partitioned into static, non-overlapping partitions. Then, for each partition, a sparse index is built on the temporal dimension. Thus, one main advantage of SETI is that it can be built on top of an existing spatial index, such as an R-tree. Furthermore, an in-memory structure is used to speed up insertions. Queries are evaluated by first performing spatial filtering, and then temporal filtering. That is, first, the candidate cells, i.e., those overlapping with the spatial range in the query, are selected. Then, for each cell, the temporal index is used to retrieve those disk pages whose timespans overlap with the temporal range in the query. Query execution concludes with a refinement step to filter out candidates and, if trajectories are desired as the output, a duplicate elimination step to filter out segments that belong to the same trajectory. A similar approach is followed also by the MTSB-tree [182] and the CSE-tree [159], which, as in SETI, partition the space into disjoint cells, but differ in the type of temporal index maintained for each cell.

An alternative approach is followed by the PA-tree [132], which instead divides first the temporal dimension into disjoint time intervals. Then, the trajectory of each object is split into a series of segments, according to these time intervals. Each segment is approximated with a single continuous Chebyshev polynomial and a two-level index is used to index these approximated trajectory segments within each time interval.

In a different direction, spatio-temporal indexes have also been proposed for indexing objects moving in a fixed network [66][44][101] or in symbolic indoor spaces [93].

3.3 Model and Definitions

Let $\mathcal{O}$ be a set of moving objects associated with a set $\mathcal{T}$ of trajectories. Each trajectory $T \in \mathcal{T}$ is approximated by a series of line segments; thus, it is defined as a tuple $T = (o, \langle \ell_1, \ell_2, \ldots, \ell_n \rangle)$, where $o \in \mathcal{O}$ is the object the trajectory belongs to and $\langle \ell_1, \ell_2, \ldots, \ell_n \rangle$ is a sequence of line segments. Each line segment is defined by a tuple $\ell = (p_s, p_e, \psi)$, where $p_s = (x_s, y_s, t_s)$ and $p_e = (x_e, y_e, t_e)$ are its start and end points, respectively, specified by a location and a timestamp, and $\psi = \{k_1, k_2, \ldots, k_m\}$ is a set containing zero or more keywords associated with this part of the trajectory. We use the notation $\ell.loc$, $\ell.\tau$, and $\ell.\psi$ to refer, respectively, to the location, the timespan, and the set of keywords of the trajectory segment ℓ.

We define the Spatio-Temporal Keyword (STK) query as a boolean range query that comprises a spatial, a temporal, and a keyword filter, i.e., $Q = (R, T, \Psi)$, where $R = [(x_s, y_s), (x_e, y_e)]$ specifies a spatial range, $T = [t_s, t_e]$ a time interval, and $\Psi = \{k_1, k_2, \ldots, k_n\}$ a set of keywords. An object $o \in \mathcal{O}$ is an answer to the STK query Q, if it has a set of (not necessarily consecutive) trajectory segments such that (a) each segment satisfies the spatial and temporal predicate of the query, and (b) the union of the keywords appearing in these segments contains all the keywords in the query. We define this more formally below.

Definition 3.1 (STK query). *Given a set of objects $\mathcal{O}$ and their trajectories $\mathcal{T}$, an STK query $Q = (R, T, \Psi)$ returns the set of objects $O \subseteq \mathcal{O}$ such that each $o \in O$ contains a set of trajectory segments $T_{o,q} \subseteq T_o$ that satisfy all of the following conditions:*

(a) $\forall \ell \in T_{o,q} : \ell.loc \cap R \neq \emptyset$

(b) $\forall \ell \in T_{o,q} : \ell.\tau \cap T \neq \emptyset$

(c) $\bigcup_{\ell \in T_{o,q}} \ell.\psi \supseteq \Psi$.

Note that in many applications, e.g., a mobile user uploading photos or posting tweets, the location updates sent by the object may be sparse. In those cases, it is not feasible to approximate the object's movement by a series of line segments and to associate relevant information, such as status or keywords, to parts of the trajectory. Instead, the information regarding location, time, and relevant keywords can only be associated to the point of each update. Nevertheless, these cases can also be addressed by the data model and query definition described above, by trivially representing each point update as a trajectory segment with zero length and duration.

3.4 Methodology

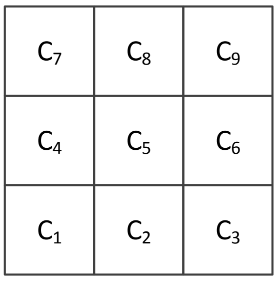
In this section, we propose two indexes for the efficient evaluation of STK queries. The first, denoted as GKR (Grid and KR*-tree), is based on a spatio-temporal index (in particular, SETI [28]) for indexing trajectories of moving objects, and extends it to incorporate the keyword information associated with the trajectory segments. The second, denoted as IFST (Inverted File with Spatio-Temporal order), is based on a spatio-textual index (in particular, SFC-QUAD [38]), and extends it to incorporate the temporal dimension.

3.4.1 The GKR Index

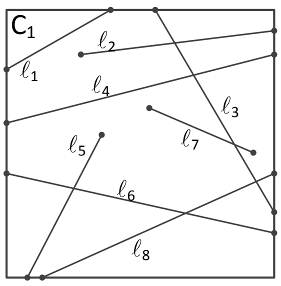
Index Description

GKR is a hybrid index that combines concepts from the SETI index [28], which has been proposed for indexing trajectories of moving objects, and the KR*-tree [83], which has been proposed for indexing spatio-textual objects.

A brief explanation of the SETI index was already provided in Section 3.2. Here, we give an overview of the KR*-tree index for spatio-textual query processing. The



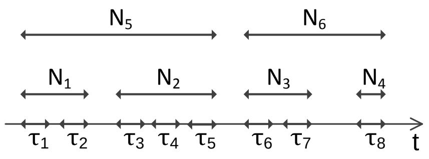
(a) Space partitioning



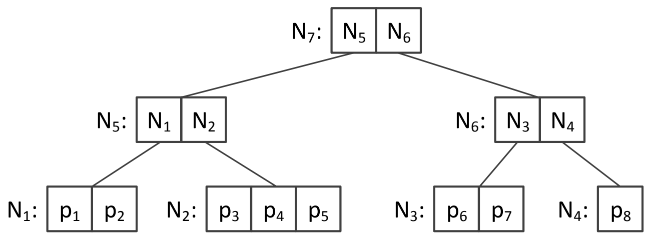
(b) Cell contents

page	segment	timespan	keywords
p_1	ℓ_1	τ_1	k_1, k_3, k_5
p_2	ℓ_2	τ_2	k_2, k_3, k_5
p_3	ℓ_3	τ_3	k_1, k_4, k_5
p_4	ℓ_4	τ_4	k_3, k_4, k_5
p_5	ℓ_5	τ_5	k_1, k_3
p_6	ℓ_6	τ_6	k_2, k_3
p_7	ℓ_7	τ_7	k_1, k_2, k_3, k_4
p_8	ℓ_8	τ_8	k_2, k_3, k_5

(c) Contained segments



(d) Segment timespans



(e) KR*-tree on cell pages

keyword	nodes
k_1	N_1, N_2, N_3, N_5, N_6
k_2	N_1, N_3, N_4, N_5, N_6
k_3	$N_1, N_2, N_3, N_4, N_5, N_6$
k_4	N_2, N_3, N_5, N_6
k_5	N_1, N_2, N_4, N_5, N_6

(f) KR*-tree list

Fig. 3.1 Example for the GKR index.

KR*-tree [83] combines R*-tree with inverted files and can be used for evaluating the boolean range spatial keyword query. The KR*-tree maintains an inverted index-like structure, called KR*-tree List, which, for each keyword, keeps a list of nodes in the R*-tree that have the keyword. This allows it to prune the search space on both spatial and textual dimensions during query processing. Each leaf node additionally contains inverted lists that index the keywords appearing in the objects under the node.

Thus, first, GKR comprises a grid which is used to spatially partition the space into a number of disjoint cells of equal size. Each cell indexes the trajectory segments that lie within it. Segments that cross multiple cells are split into parts at the points of intersection with the cell boundaries, so that each new segment is fully contained within a single cell. These newly created segments inherit the keywords of the original segment. These segments are then stored in one or more disk pages, such that each disk page only contains segments belonging to the same cell. This part is similar to SETI, which also partitions the space into disjoint cells and stores their contents in separate disk pages. However, SETI only deals with spatiotemporal data. Hence, each created disk page is characterized by its timespan, which is the union of the timespans of the segments stored in it. Then, the timespans of all pages belonging to the same cell are organized in a one-dimensional R*-tree.

Instead, in our case, each segment contained in a cell is characterized by both its timespan and the list of keywords associated with it. To deal with both dimensions, we organize the corresponding disk pages of a cell using a `KR*-tree`. Now, each disk page is associated both with its timespan, which is again the union of the timespans of each trajectory segment it contains, and with a set of keywords, which, similarly, is the union of the sets of keywords associated with its segments. The `KR*-tree` is an augmented R*-tree that additionally associates nodes with keywords. Thus, the disk pages are again organized in an R*-tree according to their timespans, but in addition a structure is maintained associating tree nodes with keywords contained in the corresponding disk pages.

Example 1. *An example illustrating the structure of the `GKR` index is shown in Figure 3.1. A grid is used to partition the space (Fig. 3.1(a)). Fig. 3.1(b) shows an example of the segments contained in the grid cell C_1. The timespans and keywords associated with these segments are shown in Fig. 3.1(c) and (d). For simplicity, in the example, we assume that each disk page stores a single segment. Fig. 3.1(e) and (f) show the KR*-tree and the KR*-tree list built according to the timespans and the keywords associated with the pages storing the segments of the cell C_1. Furthermore, each of the leaf nodes N_1, N_2, N_3, and N_4 themselves contain an inverted list each, which maps keywords to objects (in our case, disk pages) under the node containing those keywords.*

Index Construction

Since the `GKR` index combines and adapts parts from `SETI` and `KR*-tree`, the insert and update procedures also follow steps similar to the corresponding ones for those indexes. Specifically, inserting a new trajectory segment ℓ is performed following the steps described below.

a. First, the process identifies the grid cells that ℓ crosses. If there are more than one such cells, ℓ is split into multiple segments as described above, and the subsequent steps are applied for each resulting segment.

b. Next, the disk page(s) associated with that particular cell have to be checked in order to insert the new segment. In these pages, the segments are ordered chronologically, according to the timestamp of their endpoint. Thus, the `KR*-tree` associated with the cell is traversed to find the page in which the new segment should be inserted.

c. If such a page exists and is not full, the new segment is inserted. Otherwise, the contents of the subsequent pages have to be shifted or a new page has to be created.

d. Finally, the timespan and the keyword set of the affected page(s) are updated accordingly, which is then reflected in the `KR*-tree`.

We assume that in practice the `GKR` index is constructed in bulk mode, inserting trajectory segments in chronological order. Hence, each new segment is appended at the end of the last disk page of the corresponding cell (or in a new disk page, if the last one is full).

Query Evaluation

Next, we describe the steps for evaluating an `STK` query $Q = (R, T, \Psi)$ using the `GKR` index. The pseudocode for the process is presented in Algorithm 3.1.

a. First, the spatial predicate R is evaluated, by selecting all the grid cells that overlap with it. This results in a list of candidate cells. (*line* 3)

b. Then, for each candidate cell, the corresponding `KR*-tree` is traversed. The traversal identifies those nodes that: (a) have a timespan that overlaps with T, and (b) have a keyword that is contained in Ψ. From the leaf nodes reached, a set of candidate disk pages is retrieved. These pages provide a set of candidate trajectory segments that potentially satisfy predicates R and T, and contain one or more keywords from Ψ. Then, two filtering steps are applied. (*lines* 4–10)

c. The first filtering step is a refinement step in the spatial and temporal dimensions. It discards segments that are false positives, i.e., are located outside R or their timespan is outside T. This results in a set of candidate objects, which are the objects to which the remaining segments belong. (*lines* 13–17)

d. Finally, the second filtering step is applied to discard those objects whose trajectory segments from Step 3 do not fully cover the set of query keywords Ψ. (*lines* 18–20)

Algorithm 3.1: GKR Query Evaluation

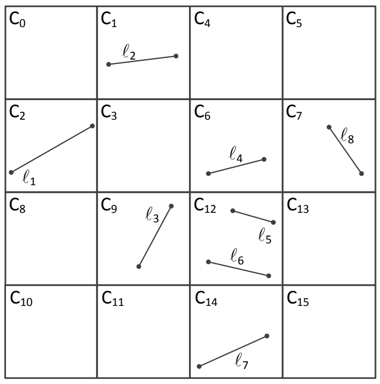
```
   Input: GKR index $I$, STK query $Q$
   Output: Set of objects $O$ satisfying $Q$
1  $O \leftarrow \emptyset$
2  $P \leftarrow \emptyset$                                        ▷ disk pages to read
3  $C \leftarrow GridCells(I) \cap Q.R$
4  foreach $c \in C$ do
5      $I_c \leftarrow$ KR*-tree associated with $c$
6      $N \leftarrow Traverse(I_c, Q.T, Q.\Psi)$
7      foreach $n \in N$ do
8          $P_c \leftarrow LoadDiskPages(N)$
9          $P_c \leftarrow FilterPages(P_c, Q.R, Q.T, Q.\Psi)$
10     $P \leftarrow P \cup P_c$
11 $L \leftarrow \emptyset$                                        ▷ candidate trajectory segments
12 $M \leftarrow \emptyset$                                        ▷ map objects to keywords
13 foreach $p \in P$ do
14     $L \leftarrow L \cup FilterSegments(p, Q.R, Q.T, Q.\Psi)$
15 foreach $\ell \in L$ do
16     $o \leftarrow$ object of $\ell$
17     $M(o) \leftarrow M(o) \cup \ell.\Psi$
18 foreach $o \in M$ do
19     if $Q.\Psi \subseteq M(o)$ then
20         $O \leftarrow O \cup o$
21 return $O$
```

3.4.2 The IFST Index

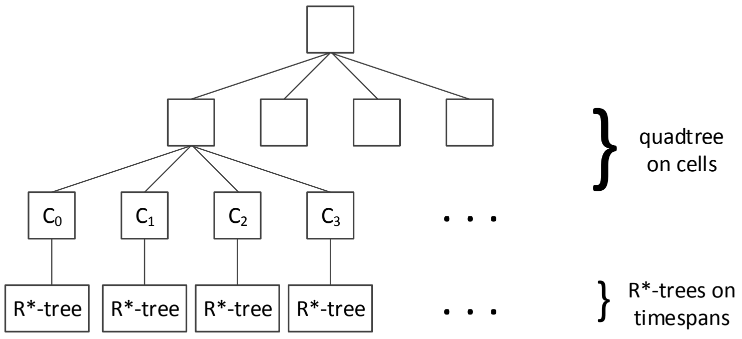
The rationale of the GKR index presented above is to enhance a spatio-temporal index with additional structures for indexing the textual dimension. Next, we follow a different direction, using a spatio-textual index as a basis and enhancing it to incorporate the temporal dimension. In particular, we describe the IFST index, which is based on SFC-QUAD [38] (see Section 2.1 for an overview), used for indexing spatio-textual objects.

Index Description

IFST comprises two main structures. The first is a global inverted file, containing, for each keyword, an inverted list with the ids of the trajectory segments that contain it. When a query is evaluated, only segments appearing in the inverted lists associated with keywords contained in the query need to be examined. Since these lists can



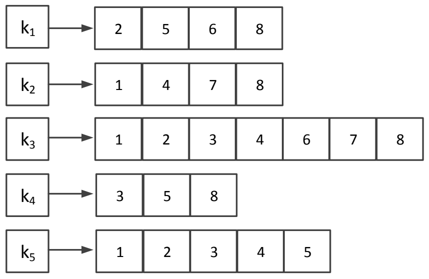
(a) Z-curve ordering of cells



(b) Spatio-temporal index

segment	cell id	segment id
ℓ_1	C_2	2
ℓ_2	C_1	1
ℓ_3	C_9	5
ℓ_4	C_6	3
ℓ_5	C_{12}	6
ℓ_6	C_{12}	7
ℓ_7	C_{14}	8
ℓ_8	C_7	4

(c) Assignment of segment ids



(d) Global inverted index

Fig. 3.2 Example for the IFST index.

still be quite long, the key issue for efficiency is to restrict the portions of each list that may contain segments satisfying the spatial and temporal predicates of the query. This is achieved by assigning ids to segments in a spatio-temporal order. For this purpose, a second, hybrid structure is maintained, which comprises in turn two parts. The first is a quadtree that indexes trajectory segments according to the spatial dimension. This allows for ordering cells, and their corresponding trajectory segments, according to a Z curve [143], so that segments that are spatially close together will also have similar ids. Furthermore, segments belonging to the same cell are assigned ids chronologically, according to the end timestamp of each segment. Then, for each leaf node of the quadtree, an R*-tree is built to index the timespans of the contained segments. Lastly, the inverted lists are themselves split into blocks (with size that is typically a multiple of 128 bytes) and are compressed using a block compression algorithm before being stored on disk.

Example 2. *An example illustrating the structure of the IFST index is shown in Fig. 3.2. A grid is used to partition the space. The cells of the grid are assigned ids according to the Z-order, as shown in Fig. 3.2(a). The cells are indexed by a quadtree; furthermore, for each leaf node, i.e., cell, an R*-tree is built on the timespans of the contained segments (Fig. 3.2(b)). We assume again 8 segments, with locations as shown in Fig. 3.2(a) and* *with timespans and keywords as in Example 1. Fig. 3.2(c) shows the new id assigned to each segment based on the Z-order of the parent cell and its chronological order within that cell. Finally, a global inverted file is built on the keywords, where the segments in each postings list are ordered according to their assigned ids (Fig. 3.2(d)).*

Index Construction

We assume that the index is constructed by inserting data in a bulk mode. The steps are described below.

a. First, a quadtree is constructed to partition the space and assign ids to cells according to the Z-order. Each trajectory segment is assigned to the corresponding cell; if it spans more than one cell, it is split as has been described previously.

b. The segments are assigned ids according to the position of their parent cell in the Z-ordered quadtree and their position in the chronological order of segments within the cell.

c. For each cell, an R*-tree is constructed to index the timespans of the contained segments.

d. Finally, an inverted file is constructed to index segments according to their keywords, using the ids assigned previously based on the spatial and temporal ordering.

Query Evaluation

The steps for evaluating an `STK` query $Q = (R, T, \Psi)$ using the `IFST` index are described below. The pseudocode for the process is presented in Algorithm 3.2.

a. The quadtree is traversed to find the leaf nodes that overlap with the spatial predicate R. (*line* 3)

b. For each leaf node, the traversal continues using the associated R*-tree to identify subsets of the contained segments that also have a timespan overlapping with T. The result is a list of segment ids, which are merged into a smaller set of k disk sweeps. (*lines* 5–8)

c. The inverted index is used to identify the posting lists for the keywords contained in Ψ. The compressed blocks corresponding to the segment ids are

Algorithm 3.2: `IFST` Query Evaluation

```
   Input: IFST index $I$, STK query $Q$
   Output: Set of objects $O$ satisfying $Q$
1  $O \leftarrow \emptyset$
2  $I_{quad} \leftarrow$ the quadtree of $I$
3  $N \leftarrow Traverse(I_{quad}, Q.R)$
4  $N' \leftarrow \emptyset$
5  foreach $n \in N$ do
6      $I_n \leftarrow$ the R*-tree associated with $n$
7      $N' \leftarrow N' \cup Traverse(I_n, Q.T)$
8  $K \leftarrow GetSegmentIdRanges(N', k)$
9  $I_{inv} \leftarrow$ the inverted index of $I$
10 $L \leftarrow \emptyset$
11 foreach $\psi \in Q.\Psi$ do
12     $L \leftarrow L \cup LoadPostings(I_{inv}(\psi), K)$
13 $M \leftarrow \emptyset$
14 foreach $\ell \in L$ do
15     if $\ell.loc \cap Q.R \neq \emptyset$ AND $\ell.\tau \cap Q.T \neq \emptyset$ then
16         $o \leftarrow$ object of $\ell$
17         $M(o) \leftarrow M(o) \cup \ell.\Psi$
18 foreach $o \in M$ do
19     if $Q.\Psi \subseteq M(o)$ then
20         $O \leftarrow O \cup o$
21 return $O$
```

read from disk in k disk sweeps and are decompressed. The result is a set of candidate trajectory segments that potentially overlap with R and T, and contain at least one of the keywords in Ψ. Then, as with `GKR`, two filtering steps are applied to obtain the result. (*lines* 11–12)

d. In the first filtering step, using document-at-a-time (DAAT) processing on the set of segment ids from the R*-trees and the set from the inverted index, the segments that are located outside R or that have their timespan outside T are discarded. This results in a set of candidate objects. (*lines* 14–17)

e. In the second filtering step, it is checked for each remaining object whether its segments from Step 4 fully cover the set of query keywords Ψ. (*lines* 18–20)

Dataset	**Size**	**Objects**	**Points**	**Keywords**
Yachts	26 MB	1,496	215,937	51,542
Flickr-EU	195 MB	46,016	1000,000	482,561
Flickr-US	189 MB	36,107	1000,000	309,839

Table 3.1 Datasets used in the experiments.

Dataset	N	$R(km^2)$	$T(hrs)$	$\lvert\Psi\rvert$
Yachts	[**216K**]	[50K, 250K, **500K**, 750K, 1M]	[6, 9, **12**, 18, 24]	[1, 2, **3**, 4, 5]
Flickr-EU	[600K, 700K, **800K**, 900K, 1M]	[1K, **5K**, 10K, 15K, 20K]	[2, 4, **6**, 9, 12]	[1, 2, **3**, 4, 5]
Flickr-US	[600K, 700K, **800K**, 900K, 1M]	[1K, **5K**, 10K, 15K, 20K]	[2, 4, **6**, 9, 12]	[1, 2, **3**, 4, 5]

Table 3.2 Parameters used in the experiments.

3.5 Experimental Evaluation

We have conducted an experimental evaluation to evaluate and compare the performance of the proposed indexes, using real-world datasets from two different sources. We first present the experimental setup, including the datasets used, and then we report the results of our experiments.

3.5.1 Datasets

In our evaluation, we used three real-world datasets coming from two different sources. These sources involve diverse types of objects, with different characteristics regarding the type of movement and keywords involved, thus allowing to test and compare our methods in diverse scenarios.

Yachts

The first is a yacht movement dataset collected from an online yacht tracking service[1]. This service allows yacht owners to register their vessels and submit GPS traces along with other information and messages. For each yacht, some basic information is provided, such as the name of the skipper and the crew, the type of the ship, and the country in which it is registered. Location updates include the coordinates, the timestamp, and optionally a short message. These short messages can be very diverse, ranging from information about the weather or the destination to various comments about the journey or the mood of the crew.

[1]http://www.yachttrack.org/

We have constructed a dataset by monitoring the service for a period of about four weeks and collecting information about past and current updates. This resulted in a dataset that contains information and location updates for 1,496 vessels around the world. It consists of approximately 216,000 points over the time period from December 2009 till June 2015. Each point is associated with latitude and longitude, a timestamp, and a set of keywords. These keywords include both metadata from the basic information about the ship, as mentioned above, and terms extracted from the message broadcast along with the respective location update.

A basic preprocessing step was applied to remove stopwords and special characters, which resulted in a set of 51,542 distinct words. Moreover, a preprocessing step has been applied on the location updates for each vessel to construct the corresponding trajectories. Specifically, we connect successive location updates of a ship to form a trajectory, if two consecutive updates are within a time period that does not exceed 30 mins. Otherwise, we split the trajectory into two different trajectories. In addition, to avoid unreasonably long segments produced due to outliers, we impose a maximum velocity restriction (100 meters per second). If the velocity calculated for the movement between two consecutive points violates this threshold, we again split the trajectory into two different trajectories. If a point is not connected to any other points, it is treated as a zero-length and zero-duration segment.

Note that this is an indicative example among many other similar services tracking the movement of aircrafts, ships, taxis, or other vehicles (e.g., Flightradar24[2], MarineTraffic[3]).

Flickr-EU/-US

The other two datasets used in the experiments are derived from the Flickr dataset provided by Yahoo [153]. This dataset contains about 99.3 million images, about 49 million of which are geotagged. For our experiments, we filtered out images that do not contain coordinates, timestamp, or tags. Then, we derived two datasets, denoted as Flickr-EU and Flickr-US, by selecting those images that are located within a bounding box around Europe and US, respectively, and also having a date in the years from 2000 to 2010. From the resulting datasets we picked 1 million photos each, belonging to 46,016 and 36,107 users, respectively.

To experiment with different dataset sizes, we also varied the number of images from 600K to 1M in each of the two datasets, using 800K as default. Again, we applied

[2]http://www.flightradar24.com/
[3]http://www.marinetraffic.com/

a preprocessing step, as described above, to construct user trajectories, treating the remaining single points as segments of zero length and duration. This data source provides an indicative case for many similar scenarios involving user generated spatio-textual data from mobile users (e.g., tweets and check-ins).

Table 3.1 shows for each of the datasets used in the experiments the total size, the total number of objects and points, and the total number of distinct keywords.

3.5.2 Performance Measures and Parameters

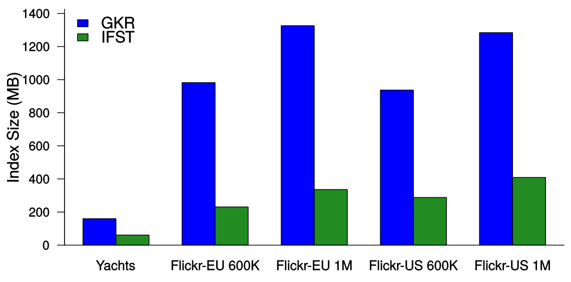
The purpose of the experimental evaluation was to compare the performance of the two proposed indexes. In particular, we focus on the following aspects: (a) index size, (b) index construction time, and (c) query evaluation time. Moreover, we examine the effect of the following parameters: (a) dataset size N (measured as total number of points), (b) size of query spatial range R, (c) length of query time interval T, and (d) number of query keywords Ψ. For each experiment, we vary the value of the selected parameter, while setting the rest parameters to their default values, as shown in Table 3.2 (default values are shown in bold).

All algorithms were implemented in Java and run on a machine with 2.8GHz Intel® Quad Core™CPU and 8GB RAM. The disk page size used was 4KB. To ensure the same spatial resolution in both the indexes, the grid resolution in GKR has been set to 64*64 and the maximum leaf node size in IFST quadtree is 50,000 segments for both Flickr-EU and Flickr-US datasets. For the Yachts dataset, these values are 64*64 and 10,000 segments per leaf node, respectively. The results of the evaluation are presented next.

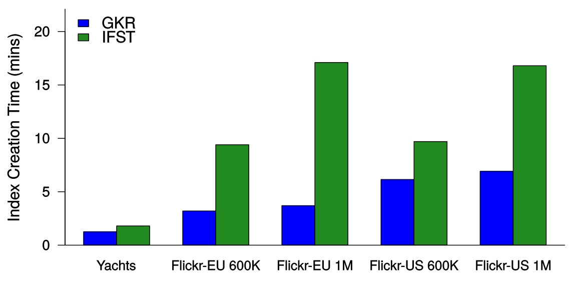
3.5.3 Index Size and Construction Time

Index Size

We start by comparing the size of the two proposed indexes. Figure 3.3(a) compares the sizes of GKR and IFST for Yachts, Flickr-EU, and Flickr-US. For the Flickr datasets, to test the increase in index size w.r.t. the dataset size, we have measured the index size for both 600K and 1M points. It can be noticed that the IFST index is much smaller in size than the GKR index. IFST achieves this space efficiency by dividing the inverted lists for each term into blocks of size 128 bytes and compressing these using a block compression algorithm. The GKR index, on the other hand, occupies around four times the size needed by IFST across all datasets. This factor is particularly



(a) Index size



(b) Index creation time

Fig. 3.3 Index size and index creation time for GKR and IFST.

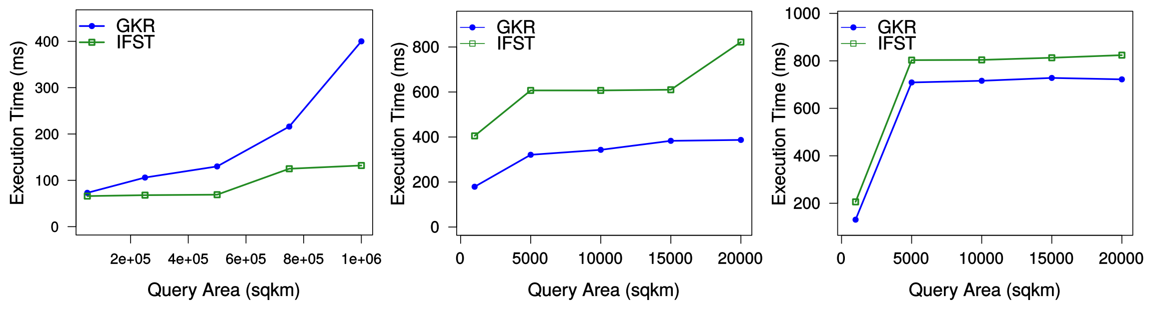
important in domains, such as Geographic Information Retrieval, where the amount of data and the number of words per object are large, and thus disk space has to be considered.

Index Construction Time

The time required to construct the indexes is depicted in Figure 3.3(b). Again, for comparison, we have chosen to show the time measurements for two dataset sizes for the Flickr datasets. It is clear from the figure that across all the datasets, the construction time of IFST is higher than that of GKR. This is due to the overhead caused by the division of the inverted lists into blocks and the compression of the blocks before being written to disk. However, this extra time spent is compensated by the savings achieved in disk space, as discussed earlier. Moreover, since we are dealing with historical trajectories of objects, index construction is often a one-time process done offline after the collection and preprocessing of data. Thus, it is usually not critical to the performance of the system.

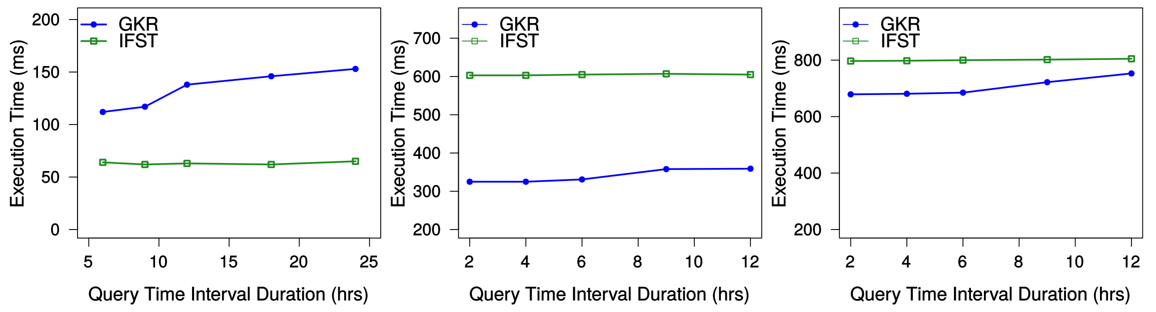
3.5.4 Query Execution Time

We now move on to a comparison of the performance of the proposed indexes in terms of their query evaluation time. We vary the query parameters, i.e, the size of the region R, the length of the time interval T, and the number of keywords Ψ in the query, and we measure the execution times. Moreover, we also generate datasets of different sizes by varying the number of points in the Flickr-EU and Flickr-US datasets by 100K from 600K to 1M. We use these to compare the scalability of the two indexes.



(a) Yachts (b) Flickr-EU (c) Flickr-US

Fig. 3.4 Execution time vs. query region size.

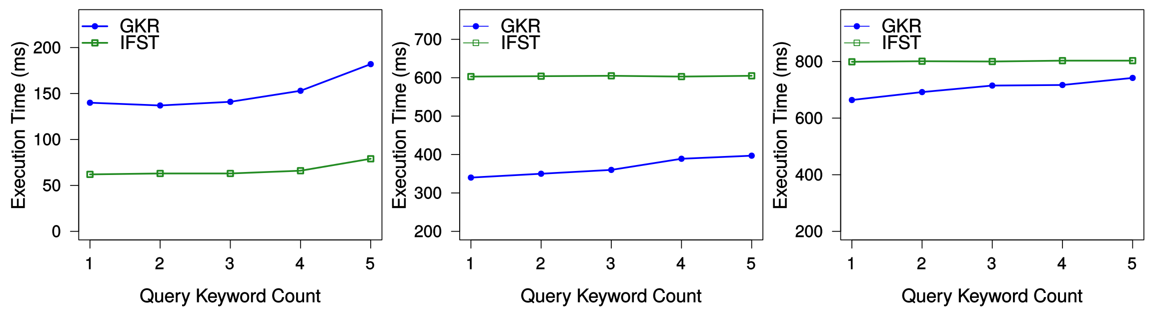


(a) Yachts (b) Flickr-EU (c) Flickr-US

Fig. 3.5 Execution time vs. query time interval.

Size of query region

The results of our experiments w.r.t. the size of the query area are shown in Figure 3.4. As can be seen, the GKR index shows superior results with Flickr-EU and Flickr-US, whereas IFST demonstrates lower query evaluation time for the Yachts dataset. The reason for this varying behavior can be attributed to the dataset size, and is discussed later. The results also show that the query execution time of both indexes tends to increase as the query area increases. This trend is due to the fact that as the query area grows, the number of objects in the dataset that intersect the query also becomes higher. The increase in times is however more marked between certain points, for example, when the query area increases from 1000 sq km to 5000 sq km in the Flickr-EU and Flickr-US datasets. The reason for this is a sudden increase in the number of grid cells in GKR and the number of leaf nodes in IFST intersecting the query as the query region grows. More number of cells and leaf nodes causes a jump in the range of segments that have to be considered and also increases the number of temporal KR*- and temporal R*-trees that have to be loaded from disk and searched.



(a) Yachts (b) Flickr-EU (c) Flickr-US

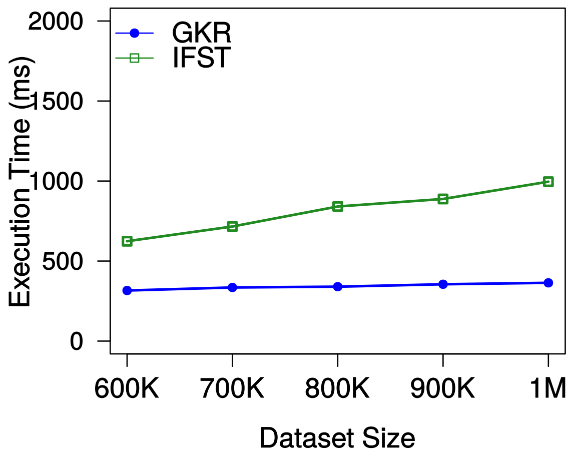
Fig. 3.6 Execution time vs. number of query keywords.

Query Time Interval

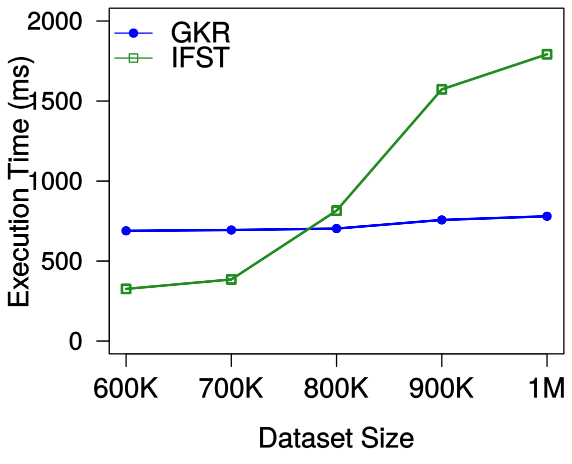
As shown in Figure 3.5, an increase in the duration of the query time interval also leads to a general increase in the query execution time for both the indexes. However, this increase is less pronounced than that produced by increasing the query area, because both indexes use spatial indexes to first filter out data which lies completely outside the query range. Again, in these experiments, `GKR` outperforms `IFST` on the Flickr datasets, while the latter performs better with the Yachts dataset. It is also noteworthy that in comparison to `GKR`, the query execution time for `IFST` varies very little with variation in the time interval duration. This is because during the refinement step, `IFST` uses DAAT processing to find the intersection of the list of segments obtained by querying the temporal R*-trees and those containing at least one query keyword read from the inverted index. On the other hand, in `GKR` a longer time interval can produce more number of disk pages whose timespans intersect the query time interval. These disk pages then have to be loaded into memory and have their segments scanned iteratively.

Number of Query Keywords

The last query parameter under consideration in our experiments is the number of query keywords. Figure 3.6 shows the query execution time as the number of keywords varies from 1 to 5. As can be noted again, the execution time generally demonstrates an upward trend with an increase in the number of keywords and the performance of `GKR` is better than that of `IFST` with the Flickr datasets, while being worse with the Yachts dataset. Also, varying the number of keywords in the query tends to impact `IFST` less than `GKR`. This is again due to DAAT processing in `IFST`, which always chooses the shorter list to iterate over during refinement and performs



(a) Flickr-EU



(b) Flickr-US

Fig. 3.7 Execution time vs. dataset size.

lookups on the longer list. In case of GKR, while querying the cell KR*-trees, more number of keywords in the query can produce more disk pages that have to be read from disk as all pages containing at least one query keyword and with timespans intersecting the query time interval have to be considered for refinement.

Dataset Size

Finally, we compare the scalability of the two indexes by experimenting with different dataset sizes. Figure 3.7 shows the results. The GKR index predominantly performs better than the IFST index and is also less sensitive to changes in the dataset size. In contrast, the query evaluation time of IFST increases significantly as the size of the dataset is increased. In fact, for smaller dataset size, its performance is close to or even better than that of GKR, as shown in Figures 3.7 and 3.4(a), 3.5(a), and 3.6(a). The reason for this behavior is that with increase in the quantity of data, the depth of the IFST quadtree also increases as nodes split further to accommodate more objects. As a consequence, the number of leaf nodes that intersect the query region becomes higher, thereby generating the overhead of loading and querying more number of R*-trees. This also makes the performance of IFST sensitive to the distribution of data, as a skewed distribution can affect its performance.

3.6 Summary

In this chapter, we have addressed the problem of efficient evaluation of spatio-temporal keyword queries on historical trajectories of moving objects. We have presented two hybrid indexes, GKR and IFST, for this purpose. The first is based on a spatio-temporal index, extended to incorporate the keywords associated with the trajectory segments. The second uses a spatio-textual index as a basis, extending it to incorporate the temporal dimension.

We have evaluated the two approaches by conducting a set of experiments using two diverse types of data, including yacht movement tracking data and geotagged images from Flickr. The results of our evaluation have shown that in terms of query evaluation time the GKR index performs better in the majority of our experiments with different query region sizes, time interval durations, number of keywords, and dataset sizes. However, the IFST index demonstrates more stable performance for varying query time intervals and keyword set sizes. It also requires less disk space and offers faster query processing times than GKR on smaller datasets.

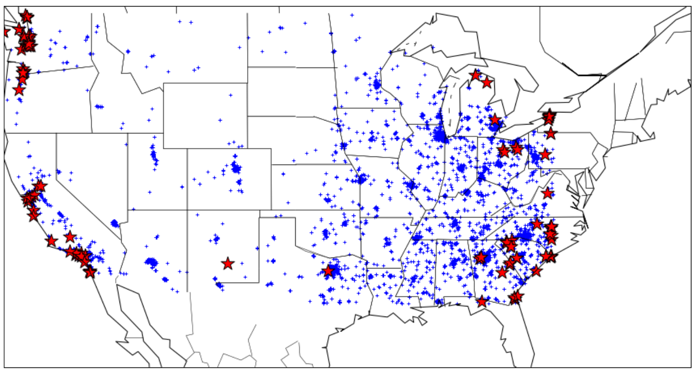
Chapter 4

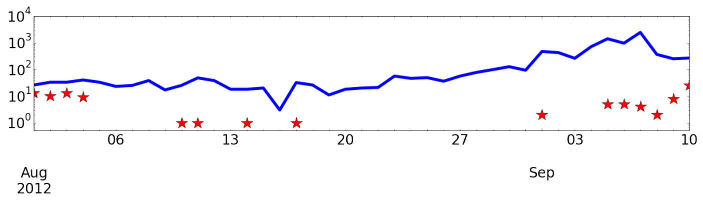
Spatial-Temporal-Textual Retrieval of Posts

The previous chapter presented the problem of spatial-temporal-textual filtering of trajectories. Now, we go a step further in this direction and study a related problem of finding a set of top-*k* posts for a given spatio-temporal range and keyword filter.

4.1 Overview

In this chapter, we address the problem of spatial-temporal-textual querying of geotagged microblog posts consisting of spatial, temporal, and textual attributes, with the goal of supporting exploratory search over topics and events with large spatio-temporal footprints. Consider a user searching microblogs for information about a topic or event. For example, the blue dots/lines in Figures 4.1(a) and 4.1(b) depict, respectively, the spatial and temporal distribution of tweets in the U.S. for a search with keywords *"obama, election"*, for a period of 40 days starting on 01/08/2012. This search returns thousands of results. Ranking results by textual relevance is often not suitable when it involves short texts or tags – essentially, every post that contains the query keyword(s) is relevant. Instead, selection and ranking of relevant posts based on their spatial and temporal attributes is much more interesting. However, both in spatio-textual and temporal information retrieval, ranking on these dimensions typically assumes that a single point in time and space is specified, so that the posts can be ranked according to their proximity to it. Nevertheless, this is challenging for topics or events that have a long span in space and time, such as those in the example, where there is not a single "central" point to use for spatio-temporal ranking. Thus, there is a need for a query type that allows for specifying a desired spatial range and time window, while still being able to retrieve top-*k* results according to spatio-temporal criteria.


(a) Spatial distribution.



(b) Temporal distribution.

Fig. 4.1 Example of results returned by a boolean query (blue) and the corresponding *k*CD-STK query (red).

To that end, we introduce a novel type of query, the *top-k Coverage and Diversity aware Spatio-Temporal Keyword* (*k*CD-STK) query. Intuitively, the goal is to return top-*k* results, where the ranking is driven by the spatio-temporal distribution of the posts. Thus, we consider as more relevant, posts that lie within dense areas in the three-dimensional spatio-temporal space. Specifically, we introduce the criterion of spatio-temporal *coverage*, which assigns a score to each post based on the number of other posts that lie within a specified distance threshold to it in the spatio-temporal dimensions. Furthermore, to avoid over-representing these areas while missing other interesting results, we also try to maximize the spatio-temporal *diversity* among the selected posts. Returning to the example presented in Figure 4.1, the red stars correspond to a subset of 100 results selected by the *k*CD-STK query. Notice that the selected results are more spread out in space and time, instead of focusing around a single area, thus better representing the whole set of relevant posts.

The *k*CD-STK query is particularly useful for data exploration, such as analysis and summarization of events and topics based on user-generated content in social media. For example, consider a large scale event for which thousands of geolocated and timestamped tweets or photos exist. The *k*CD-STK query allows to obtain a few representative documents that reveal the spatio-temporal distribution of the relevant content. In contrast, the spatial keyword queries studied in the literature, besides ignoring the temporal dimension, would either return a huge result set (boolean range query) or require the user to choose a specific point in space (and, accordingly, time) according to which the results would be ranked (boolean or top-*k* *k*NN query).

Thus, the *k*`CD-STK` query extends the standard spatial keyword queries in two main aspects: (a) it includes a temporal filter, in addition to the spatial filter, thus allowing the retrieval of documents that are associated with both a location and a timestamp; (b) instead of ranking the results by spatial proximity or spatio-textual relevance, it computes a diversified subset of *k* documents based on the introduced measures of spatio-temporal coverage and diversity. The former assigns higher weights to documents that are located in more dense parts of the spatio-temporal space. Thus, the selected results are more representative, i.e., better reflect the distribution of the whole result set. The second condition considers the pairwise spatio-temporal distance of the selected results, thus avoiding very similar results to be returned.

The *k*`CD-STK` query is founded on the basic concepts commonly used for search results diversification. In particular, it introduces the concept of coverage [52] in the search results diversification framework presented in [73] (see Section 4.2.1 for more details). By determining the relevance of each result to the query indirectly, i.e., based on the number of other results it covers, it allows the spatio-temporal filters in the query to be defined more flexibly, indicating a whole spatial region and a time window rather than requiring the user to restrict his search around a specific location and point in time. This makes the *k*`CD-STK` query more suitable for exploratory search. As the returned top-*k* results more closely reflect the spatio-temporal distribution of the whole result set, they can serve as *anchor points* for further exploration of the available posts.

Since typical diversification problems are known to be NP-hard, the challenge that arises in practice is how to efficiently evaluate a *k*`CD-STK` query, so that the results can still be retrieved in real time. The aforementioned approaches are general frameworks for results diversification, thus none of them deals particularly with the spatio-temporal coverage or diversity of posts. To the best of our knowledge, our work is the first to introduce these criteria and to consider their efficient evaluation in the context of spatial-temporal-keyword queries. More specifically, the main contributions of this chapter can be summarized as follows:

- We formally introduce a novel type of spatial-temporal-keyword query, the *k*`CD-STK` query. This query allows a keyword search to be issued with spatial and temporal range filters, and then ranks the matching results according to the criteria of spatio-temporal *coverage* and *diversity*.

- We propose an efficient strategy for evaluating a kCD-STK query. Then, we show how state-of-the-art hybrid spatio-textual indexes can be adapted and extended to be used with this strategy for efficiently selecting the top-k results from the whole set of relevant posts.
- We experimentally evaluate our approach, using two large, real world datasets containing geotagged tweets and photos. The experiments demonstrate that our approach can effectively exploit the underlying index structure, thus significantly reducing the time for computing the top-k coverage and diversity aware results.

The rest of the chapter is structured as follows. Section 4.2 provides additional background required for our problem, focusing on search results diversification and temporal keyword queries. Then, the kCD-STK query is formally introduced in Section 4.3, defining the criteria for spatio-temporal coverage and diversity. Section 4.4 presents our approach and describes how it can be applied with state-of-the-art hybrid indexes for spatial keyword queries, after extending them to include the temporal dimension. Finally, Section 4.5 presents our experimental evaluation, and Section 4.6 concludes the chapter.

4.2 Additional Relevant Background

Next, we present an overview of relevant work on search results diversification and temporal keyword queries.

4.2.1 Search Results Diversification

Typically, Web search engines only retrieve top-k results, since in most cases there are thousands or millions of documents relevant to the query. However, ranking search results purely by relevance often leads to including many similar documents in the top results, hence causing repetition and redundancy in the result set. To avoid repetition and increase the novelty of information, search results diversification has been proposed as a more advanced technique for selecting a subset of results to present to the user [73, 50, 25, 158, 3, 48, 52]. The goal is to improve the utility of the results by increasing their novelty, thus improving the user experience, especially during exploratory search. More specifically, content-based diversification aims at selecting a subset of documents that maximizes an objective function with two components: *relevance* and *diversity*. The former measures how relevant a result is for the query, while the latter measures the dissimilarity or novelty of that result w.r.t. others already selected.

Many different formulations have been proposed for search results diversification (refer to [73, 50] for classification). The most well-known approach is the framework proposed in [73]. According to it, the problem is defined as selecting a subset $\mathcal{R}^*$ of the whole result set $\mathcal{R}$, with $|\mathcal{R}^*| = k$, that maximizes an objective function ϕ, which combines the criteria of relevance and diversity. There exist different ways to define ϕ, leading to different variants of the problem. For example, in the max-sum variant, ϕ is defined as the weighted sum of two components: the total relevance of documents and the sum of pairwise distances among the documents.

As shown in [73], the max-sum problem, as well as other similar variants, is NP-hard by reduction to the MaxSumDispersion problem. Thus, greedy heuristics are used in practice to efficiently compute a diversified subset of the results. The main approach is to incrementally construct the diversified result set by choosing at each step the object that maximizes a certain scoring function. A well-known function for this purpose is the *maximal marginal relevance* (*mmr*) [25]. An evaluation of various object scoring functions and different heuristics can be found in [158].

Other types of diversification problems have also been studied, such as taxonomy or classification-based diversification [3, 157] and multi-criteria diversification [48]. Closer to our work is the coverage problem [52]. Here, the goal is to select the minimum subset of documents, such that the selected documents are diverse, i.e., have distance to each other at least ϵ, and *cover* the whole dataset, i.e., each remaining object lies within distance ϵ from a selected one. This formulation is suitable for data exploration and summarization; however, in this case the size of the selected subset is not fixed, but depends instead on the distance threshold ϵ.

In our approach, we combine the criterion of coverage from [52] with the general diversification framework of [73]. Thus, the relevance of each result is determined indirectly based on the number of other results it covers from the original set, while the number of results to return is still explicitly specified in the query. Moreover, all aforementioned works focus on formulating the diversification problem in a generic manner, using abstract definitions for document relevance and distance. Subsequently, the efficiency of computation is addressed by introducing heuristic algorithms that compute an approximation of the optimal solution. On the other hand, we focus on the specific problem of selecting spatio-temporally diverse subsets of results. We define concrete criteria for spatio-temporal coverage and diversity, and we show how an underlying index can be exploited to further speed up the computation.

4.2.2 Temporal Keyword Queries

From our previous discussion in Chapters 1 and 2, it is evident that spatial keyword queries have been studied extensively in the past few years. However, all these approaches consider only the spatial dimension of documents, thus ignoring any explicit or implicit temporal information present. Integrating temporal information into traditional web search has been the focus of research in temporal information retrieval (TIR), where a large body of work already exists (see [19, 6] for recent surveys). A main challenge in this area involves the interpretation of temporal expressions, which can vary significantly, including explicit (e.g., "January 2016"), implicit (e.g., "New Year's Eve"), and relevant ones (e.g., "last week"). With respect to indexing, a basic approach is to include the temporal attribute in the posting list [14, 7], while other works have proposed the use of a hybrid index [95] or two separate indexes [8]. However, only few works have considered both dimensions of space and time in keyword queries. Some of these, including [130], have already been discussed (see Section 2.5.1).

In a different direction, spatio-textual publish/subscribe (see Section 2.4.1) combines the criteria of textual relevance, spatial proximity, and recency to continuously maintain all or top-*k* relevant results over a stream of geo-textual documents [34]. Finally, other works in TIR have dealt with timelines and summaries of event-related information in microblogs [5, 92].

However, these approaches either apply the spatio-temporal criteria as boolean filters or use them to rank documents based on spatial proximity and/or recency. To the best of our knowledge, our work is the first to introduce the criteria of spatio-temporal coverage and diversity in keyword queries.

4.3 Model and Definitions

We now provide the basic definitions necessary to formulate the problem at hand.

Definition 4.1 (Post). *A spatio-temporal post D is represented by a tuple $D = \langle loc, t, \Psi \rangle$, where $loc = (x, y)$ are the coordinates of the location where the post was* *made, t is the timestamp of the post, and Ψ is a keyword vector containing zero or more terms, keywords, or tags contained in the post.*

Definition 4.2 (STK Filter)**.** *A spatial-temporal-keyword filter F is a tuple $F = \langle R, T, \Psi \rangle$, where the spatial filter $R = [(x_{min}, y_{min}), (x_{max}, y_{max})]$ specifies a spatial bounding box, the temporal filter $T = [t_{min}, t_{max}]$ specifies a time window, and the keyword filter $\Psi = \{\psi_1, \psi_2, \ldots, \psi_n\}$ specifies a set of keywords.*

For the remainder of this chapter, we use the dot notation to refer to a tuple's attribute values. The next definition determines when a post is considered relevant for a given STK filter.

Definition 4.3 (Relevant Posts)**.** *Given a collection $\mathcal{D}$ of posts and an STK filter F, the set of relevant posts $\mathcal{D}_F$ contains all posts $D \in \mathcal{D}$ such that (i) $D.loc \in F.R$, (ii) $D.t \in F.T$, and either (iii-a) $D.\Psi \cap F.\Psi \neq \emptyset$ under OR semantics, or (iii-b) $D.\Psi \supseteq F.\Psi$ under AND semantics.*

Notice that the difference between OR and AND semantics is whether a relevant post must contain *at least one* or *all* of the keywords that appear in the filter.

As discussed in Section 4.1, for the type of posts and STK filters that motivate our work, i.e., exploratory search for topics or events that are distributed across potentially large intervals in space and time, the number of relevant posts is typically very high. Therefore, our objective is to select a small subset of k relevant posts that have high *coverage* and *diversity*. To elaborate on these two notions, we first need to introduce measures of spatial and temporal distance between two relevant posts (w.r.t. an STK filter F) $D_i, D_j \in \mathcal{D}_F$.

The spatial distance is defined as:

$$d_s(D_i, D_j) = \frac{d(D_i.loc, D_j.loc)}{\sigma_{max}},$$

where $d((x,y),(x',y'))$ is the Euclidean distance between two points and σ_{max} is a normalization factor corresponding to the length of the diagonal of $F.R$, i.e., the maximum possible spatial distance between any pair of posts lying in $F.R$. Note that it is possible to use other functions (e.g., L_p norms) to measure spatial distance; the changes to our methodology are straightforward.

Similarly, the temporal distance is defined as:

$$d_t(D_i, D_j) = \frac{|D_i.t - D_j.t|}{\tau_{max}},$$

where $\tau_{max} = F.t_{max} - F.t_{min}$ is a normalization factor corresponding to the maximum possible temporal distance. As before, one could also employ other functions for the temporal distance, e.g., to assign greater importance to more recent posts.

We are now ready to introduce our two key notions, *coverage* and *diversity*. We first define them for individual posts, and then extend the definitions to sets of posts.

Definition 4.4 (Coverage). *Given a collection $\mathcal{D}$ of posts and an STK filter F, the coverage of a post $D \in \mathcal{D}_F$ is:*

$$cov(D) = \frac{1}{|\mathcal{D}_F|}|\{D' \in \mathcal{D}_F : d_s(D,D') \leq \rho_s \ \& \ d_t(D,D') \leq \rho_t\}|, \tag{4.1}$$

where $\rho_s, \rho_t \in [0,1]$ are unit-less spatial and temporal distance thresholds, respectively. Moreover, the coverage of a set of posts $\mathcal{R} \subseteq \mathcal{D}_F$ of size k is:

$$cov(\mathcal{R}) = \frac{1}{k}\sum_{D \in \mathcal{R}} cov(D). \tag{4.2}$$

Since each post in the set $\mathcal{R}$ can potentially cover all $|\mathcal{D}_F|$ relevant posts, the denominators in the above equations ensure that coverage takes values in the $[0,1]$ range.

Definition 4.5 (Diversity). *Given a collection $\mathcal{D}$ of posts and an STK filter F, the diversity of a pair of posts $D_i, D_j \in \mathcal{D}_F$ is:*

$$div(D_i,D_j) = w \cdot d_s(D_i,D_j) + (1-w) \cdot d_t(D_i,D_j), \tag{4.3}$$

where $w \in [0,1]$ is an application-specific weight parameter between the spatial and the temporal distances. Moreover, the diversity of a set of posts $\mathcal{R} \subseteq \mathcal{D}_F$ of size k is:

$$div(\mathcal{R}) = \frac{1}{k \cdot (k-1)}\sum_{D_i,D_j \in \mathcal{R}, i \neq j} div(D_i,D_j). \tag{4.4}$$

As there are $k \cdot (k-1)$ ordered pairs of posts in set $\mathcal{R}$, the denominator normalizes diversity in the $[0,1]$ range.

We can now define the Coverage & Diversity aware top-k STK query.

Definition 4.6 (k`CD-STK` Query). *Given a collection $\mathcal{D}$ of posts, a Coverage and Diversity aware top-k STK query specifies an STK filter F and seeks for a result set $\mathcal{R}^*$ of k relevant* *posts such that:*

$$\mathcal{R}^* = \underset{\mathcal{R} \subseteq \mathcal{D}_F, |\mathcal{R}|=k}{\arg\max} \{(1-\lambda) \cdot cov(\mathcal{R}) + \lambda \cdot div(\mathcal{R})\}, \tag{4.5}$$

where $\lambda \in [0,1]$ is a parameter determining the trade-off between coverage ($\lambda = 0$) and diversity ($\lambda = 1$).

4.4 Methodology

We now present our methodology for evaluating the k`CD-STK` query. It is split into two phases; we first determine the set of relevant posts, and then construct the result set by identifying k posts with high coverage and diversity. For each phase, we state the objective, outline the proposed approach, and then elaborate on the implementation using state-of-the-art index structures from the literature.

4.4.1 Finding Relevant Posts

For a given STK filter F, the objective of the first phase is to obtain the posts that satisfy F, assuming OR or AND semantics. Our approach is to employ existing techniques used to retrieve documents based on spatial and textual criteria, and extend them to act as filters and, more importantly, to be able to handle the temporal information. Therefore, we discuss next two distinct implementations, one based on the RCA approach [176], and another using the I^3 index [175]. For a background of the I^3 index and the RCA algorithm, refer to Section 2.1.

RCA-based Implementation

We follow the rationale of the RCA method for ranking documents based on a spatio-textual score. Recall that in this method each keyword is associated with two postings lists, one which sorts documents in descending order of textual relevance, and another which sorts documents according to their Z-order encoding of their locations. For our purposes, the first postings list can be ignored. To facilitate filtering using spatial and temporal predicates, we compute the Z-order over the 3D spatio-temporal space. As described earlier, the filtering property of the Z-order encoding states that for a spatial bounding box R, with z_{min} and z_{max} being the Z-order encodings of its top-left and bottom-right corners, respectively, the Z-order encoding of any point that lies within R has a value $z \in [z_{min}, z_{max}]$. This characteristic is used to eliminate posts that lie outside the given spatio-temporal filters.

In particular, the retrieval of relevant posts proceeds as follows.

- Determine the Z-order range $[z^-, z^+]$ that minimally covers the spatial $F.R$ and temporal $F.T$ ranges specified by the filter F.
- For each keyword ψ in the filter $F.\Psi$, retrieve from the corresponding postings list only those posts with Z-order encoding in the $[z^-, z^+]$ range.
- For each keyword, eliminate false positives, i.e., posts within the $[z^-, z^+]$ range that do not satisfy the spatial $F.R$ and temporal $F.T$ ranges. This is a necessary step given the inherent limitation of Z-order encoding [176].
- Merge the lists with the surviving posts per keyword. For OR semantics, return the union, while for AND semantics, return the intersection of the lists.

I^3-based Implementation

We employ the I^3 index and the associated methodology presented in [175] for retrieving documents based on a spatio-textual score. As with the case of the RCA-based implementation, we need to extend the underlying index structure to support retrieval using both spatial and temporal criteria. Therefore, instead of having a quadtree associated with each keyword, we construct an octree indexing documents in the 3D spatio-temporal space. Then, the retrieval of relevant documents proceeds largely similar to [175].

The algorithm is best understood by conceptualizing a single virtual (i.e., non-materialized) octree. We say that a keyword is dense for a particular cell, if the number of posts that lie within the cell and contain this keyword exceeds the disk page capacity. With each cell, we associate the set of posts that have a non-dense keyword, and for each dense keyword a signature summarizing the posts with that keyword. A cell has children cells if it has at least one dense keyword.

To find the relevant posts for F, we perform a depth-first traversal of the octree. A cell is only visited if it overlaps with the spatial $F.R$ and temporal $F.T$ ranges. In addition, a cell is pruned if it can be guaranteed that the sub-tree rooted at this cell contains no relevant posts. This check differs depending on the keyword filter semantics. For OR semantics, the cell is pruned if the associated set of posts is empty and the union of the signatures for the non-dense keywords among $F.\Psi$ is empty. For AND semantics, the cell is pruned if no associated post is contained in the intersection of the signatures for the non-dense keywords among $F.\Psi$. At the end of this traversal, the set of posts associated with all non-pruned leaf cells constitute the set of relevant posts.

4.4.2 *k*CD-STK Query Processing

Processing a *k*`CD-STK` query is a computationally hard optimization problem. Indeed, if we set parameters $\lambda = 1$ and $w = 1$ for instance, we seek for a set $\mathcal{R}$ of k posts that maximize the objective function $\sum_{i \neq j \in \mathcal{R}} d(D_i.loc, D_j.loc)$. This is precisely the 2D-MaxSumDispersion problem for which no exact, polynomial time algorithm is known (although it remains open whether 2D-MaxSumDispersion is NP-hard, similar MaxSumDispersion problems are [137]). Therefore, we turn to heuristic algorithms for constructing the result set of a *k*`CD-STK` query.

In particular, we adopt the standard greedy method for constructing the set incrementally, where at each step the document that has the highest *marginal gain* on the objective function is added. It is known that such an approach gives a 2-approximation for the general MaxSumDispersion problem [15]. In our context, the objective function for a set of posts $\mathcal{R}$ is:

$$\phi(\mathcal{R}) = (1 - \lambda) \cdot cov(\mathcal{R}) + \lambda \cdot div(\mathcal{R}),$$

and the marginal gain $g(D) \equiv \phi(\mathcal{R} \cup \{D\}) - \phi(\mathcal{R})$ for including $D \in \mathcal{D}_F \setminus \mathcal{R}$ is:

$$g(D) = \frac{1 - \lambda}{k} \cdot cov(D) + \frac{\lambda}{k \cdot (k-1)} \sum_{D_i \in R} div(D, D_i). \tag{4.6}$$

In other words, the marginal gain on the objective function of post D is the weighted sum of its coverage and its diversity to the existing posts in the set $\mathcal{R}$. In what follows, we first describe the straightforward approach of implementing the greedy algorithm, which will serve as our baseline. Then, we introduce a generic index-aware methodology that takes advantage of the underlying index structure in order to speed up the greedy algorithm.

Baseline Greedy Algorithm

Once all relevant posts have been identified, the *Baseline Greedy Algorithm*, denoted as `BSL`, directly implements the greedy heuristic for the MaxSumDispersion problem.

Algorithm 4.1: Algorithm BSL

```
Input: document collection $\mathcal{D}$, STK filter $F$, result set size $k$
Output: coverage and diversity aware result set $\mathcal{R}^*$
1 $\mathcal{D}_F \leftarrow$ FindRelevantDocs$(\mathcal{D}, F)$          ▷ Section 4.4.1
2 $\mathcal{R}^* \leftarrow \emptyset$
3 while $|\mathcal{R}^*| < k$ do
4     foreach $D \in \mathcal{D}_F$ do
5         compute $g(D)$                                  ▷ Equation 4.6
6     find document $D^*$ that maximizes $g(D^*)$
7     $\mathcal{R}^* \leftarrow \mathcal{R}^* \cup \{D^*\}$
```

Algorithm 4.1 shows the pseudocode for BSL. Initially, the set of relevant posts is retrieved (line 1), following the methodology discussed in Section 4.4.1. Then the result set is built incrementally. At each iteration (lines 3–7), the marginal gain of each post is computed by applying Equation 4.6 (line 5). The post with the highest marginal gain is selected for insertion in the result set (lines 6–7). The algorithm terminates as soon as k posts have been selected (line 3).

When computing the marginal gains, one thing to notice is that the coverage term remains fixed across all iterations for a particular post D. The reason is that $cov(D)$ depends on the fixed set $\mathcal{D}_F$ of relevant posts, rather than the partial result set. Therefore, we only need to compute this first term once for all posts.

Index-Aware Greedy Algorithm

The main drawback of the BSL algorithm is that it computes (or updates) the marginal gain for every relevant post up to k times. When the number of relevant posts $|\mathcal{D}_F|$ is large, this constitutes a performance bottleneck. It would be desirable to avoid computing the marginal gain for posts that are most likely to not be included in the result set. To achieve this goal, we propose the *Index-Aware Greedy Algorithm*, termed IDX, that takes advantage of the existing index structure to speed up kCD-STK query processing. We first overview IDX without specific assumptions on the index, and later delve into implementation details assuming explicitly an RCA or an I^3 approach. We emphasize that our methodology is generic and can be readily applied over other spatio-textual indexes (provided they can be extended to handle temporal information).

The basic idea of IDX is to form groups by clustering relevant posts that have similar spatial and temporal information. Thanks to the inherent spatio-temporal clustering of the underlying index, the groups are constructed with negligible overhead. Then, at each iteration and for each group, we compute an upper bound on its marginal gain. Groups that are promising, i.e., have a high upper bound, are examined more closely by looking at their members. On the other hand, at each iteration, unpromising groups can be dismissed, thus avoiding to compute the exact marginal gain of their members.

With each group G, we associate the following information.

- Its cardinality $|G|$.
- Its spatial extent $G.R$, which is a rectangle that minimally bounds the locations of the group's posts.
- Its temporal extent $G.T$, which is a time interval that minimally bounds the timestamps of the group's posts.
- A lower bound $G.cov^-$ on the coverage of any post in the group.
- A set $G.par$ that contains groups that are *partially covered* by G. We say that a post covers another if their spatial and temporal distances are within the spatial and temporal distance thresholds respectively. We say that a group G partially covers another G', if there can exist a post D in the former and two posts in the latter such that one is covered by D, while the other is not.
- A value $G.div^+$, which is an upper bound on the diversity of any post in the group to all posts in $\mathcal{R}$.

Based on this information, we can compute an upper bound $g(G)^+$ on the marginal gain of any member D in group G as follows:

$$g(G)^+ = \frac{1-\lambda}{k} \cdot \left(G.cov^- + \frac{1}{|\mathcal{D}_F|} \sum_{G' \in G.par} |G'| \right) + \frac{\lambda}{k \cdot (k-1)} G.div^+. \tag{4.7}$$

We next discuss how to derive the group information. To compute $G.cov^-$ and construct $G.par$, we iterate across the groups, and for each such group G', we compute spatial and temporal bounds:

$$d_s^-(G, G') = \frac{\texttt{mindist}(G.R, G'.R)}{\sigma_{max}} \text{ and } d_s^+(G, G') = \frac{\texttt{maxdist}(G.R, G'.R)}{\sigma_{max}},$$

$$d_t^-(G, G') = \frac{\texttt{mindist}(G.T, G'.T)}{\tau_{max}} \text{ and } d_t^+(G, G') = \frac{\texttt{maxdist}(G.T, G'.T)}{\tau_{max}},$$

where the `mindist` and `maxdist` are the standard functions that return the minimum and maximum possible distances respectively between ranges (Euclidean distance for spatial ranges and absolute value for temporal ranges). Intuitively, these values bound the spatial and temporal distances between any pair of posts from groups G and G'. We thus distinguish the following cases:

a. $d_s^+(G, G') \leq \rho_s$ and $d_t^+(G, G') \leq \rho_t$: we increment $G.cov^-$ by $\frac{|G'|}{|\mathcal{D}_F|}$.

b. $d_s^-(G, G') \leq \rho_s$ and $d_t^-(G, G') \leq \rho_t < d_t^+(G, G')$: we insert G' into $G.par$.

c. $d_s^-(G, G') \leq \rho_s < d_s^+(G, G')$ and $d_t^-(G, G') \leq \rho_t$: we insert G' into $G.par$.

Regarding $G.div^+$, notice that its value only increases across iterations of the greedy algorithm, as new posts are inserted in the result set $\mathcal{R}$. Therefore, at the end of an iteration, assuming D^* is inserted in $\mathcal{R}$, we update $G.div^+$ as:

$$G.div^+ \leftarrow G.div^+ + w \cdot \frac{\texttt{maxdist}(G.R, D^*.loc)}{\sigma_{max}} + (1 - w) \cdot \frac{\texttt{maxdist}(G.T, D^*.t)}{\sigma_{max}}. \tag{4.8}$$

We are now ready to present the IDX algorithm, whose pseudocode is shown in Algorithm 4.2. As in BSL, the first step is to retrieve the relevant posts using the methodology from Section 4.4.1 (line 1). Then, these are clustered into the set of groups $\mathcal{G}$ (line 2). The exact partitioning depends on the underlying index structure; we briefly discuss this later. The next step is to compute the coverage information associated with each group (lines 3–7). In particular, for each group G, the spatial and temporal bounds are computed (line 6), and the value $G.cov^-$ and the set $G.par$ are updated according to the three cases described earlier.

Subsequently, the main loop of the algorithm begins (lines 10–36), where at the end of each iteration a new post is added to the result set $\mathcal{R}^*$ until k posts are selected. Note that there are three primary data structures in IDX; the set of groups $\mathcal{G}$, the set of seen posts $\mathcal{D}_{seen}$, and the heap H which directs the examination of groups in a best-first manner. Initially, $\mathcal{G}$ contains all groups and $\mathcal{D}_{seen}$ is empty. In the heap, an entry $\langle g(G)^+, G\rangle$ for each group G is inserted, where the upper bound on the marginal gain $g(G)^+$ is the key, and is computed from Equation 4.7 (lines 12–14). Also, in the heap, an entry $\langle g(D), D\rangle$ is inserted for each post D having key its current marginal gain $g(D)$.

The inner loop (lines 17–29) examines entries from the heap H until the top entry with the highest key (corresponding to marginal gain or an upper bound thereof) belongs to a post (line 17). At that point, this entry $\langle g(D^*), D^*\rangle$ is deheaped (line

Algorithm 4.2: Algorithm `IDX`

```
   Input: document collection $\mathcal{D}$, STK filter $F$, result set size $k$
   Output: coverage and diversity aware result set $\mathcal{R}^*$
1  $\mathcal{D}_F \leftarrow$ FindRelevantDocs($\mathcal{D}, F$)                    ▷ Section 4.4.1
2  cluster $\mathcal{D}_F$ into a set of groups $\mathcal{G}$                       ▷ index dependent
3  foreach group $G \in \mathcal{G}$ do
4      $G.cov^- \leftarrow 0$; $G.par \leftarrow \emptyset$
5      foreach group $G' \in \mathcal{G}$ such that $G' \neq G$ do
6          compute bounds $d_s^-(G,G')$, $d_s^+(G,G')$, $d_t^-(G,G')$, $d_t^+(G,G')$
7          update $G.cov^-$ and $G.par$ according to the three cases
8  $\mathcal{D}_{seen} \leftarrow \emptyset$                                          ▷ set of seen posts
9  $\mathcal{R}^* \leftarrow \emptyset$
10 while $|\mathcal{R}^*| < k$ do
11     $H \leftarrow \emptyset$                                                       ▷ initialize heap
12     foreach group $G \in \mathcal{G}$ do
13         compute $g(G)^+$                                                           ▷ Equation 4.7
14         enheap in $H$ entry $\langle g(G)^+, G \rangle$
15     foreach document $D \in \mathcal{D}_{seen}$ do
16         enheap in $H$ entry $\langle g(D), D \rangle$
17     while $H.top$ is a group entry do
18         deheap from $H$ top entry $\langle g(G)^+, G \rangle$
19         $\mathcal{G} \leftarrow \mathcal{G} \setminus \{G\}$
20         foreach document $D \in G$ do
21             $\mathcal{D}_{seen} \leftarrow \mathcal{D}_{seen} \cup \{D\}$
               ▷ compute the coverage of $D$
22             $cov(D) \leftarrow G.cov^-$
23             foreach document $D' \in G' \in G.par$ do
24                 if $d_s(D,D') \leq \rho_s$ and $d_t(D,D') \leq \rho_t$ then
25                     $cov(D) \leftarrow cov(D) + \frac{1}{|\mathcal{D}_F|}$
               ▷ compute the diversity of $D$
26             $div(D) \leftarrow 0$
27             foreach document $D' \in \mathcal{R}^*$ do
28                 $div(D) \leftarrow div(D) + div(D,D')$
               ▷ compute the marginal gain of $D$
29             $g(D) = \frac{1-\lambda}{k} \cdot cov(D) + \frac{\lambda}{k \cdot (k-1)} div(D)$
30             enheap in $H$ entry $\langle g(D), D \rangle$
31     deheap from $H$ top entry $\langle g(D^*), D^* \rangle$
32     $\mathcal{R}^* \leftarrow \mathcal{R}^* \cup \{D^*\}$
33     foreach $G \in \mathcal{G}$ do
34         update $G.div^+$                                                           ▷ Equation 4.8
35     $\mathcal{D}_{seen} \leftarrow \mathcal{D}_{seen} \setminus D^*$
36     foreach $D \in \mathcal{D}_{seen}$ do
37         $g(D) \leftarrow g(D) + \frac{\lambda}{k \cdot (k-1)} div(D, D^*)$         ▷ update $g(D)$
```

Table 4.1 Datasets used in the experiments.

Dataset	Number of geotagged posts	Number of distinct keywords	Average number of keywords	Temporal coverage	Spatial coverage	Disk storage	Index size (I^3-based)	Index size (RCA-based)
Twitter	20M	1,836,679	5.7	Apr.-Dec. 2012	Worldwide	1.5GB	29GB	11GB
Flickr	20M	1,306,785	8.4	2010-2014	Worldwide	2.3GB	79GB	16GB

31) and the corresponding post is inserted in the result set (line 32). Because a new result has just been found, the information regarding the diversity of all groups (lines 33–34) and all seen posts, except D^*, (lines 35–37) is updated.

In an iteration of the inner loop (lines 17–30), where entry $\langle g(G)^+, G\rangle$ is deheaped, the following takes place. The group is removed from the set $\mathcal{G}$ of groups (line 19) and all its posts are inserted in set $\mathcal{D}_{seen}$ (line 21). Moreover, for each post $D \in \mathcal{G}$, its exact coverage $cov(D)$ (lines 22–25), its diversity $div(D)$ (lines 26–28), and ultimately its marginal gain $g(D)$ (line 29) are computed. When computing the coverage of D, its group coverage information, $G.cov^-$ and $G.par$, is used to speed up the process. Then an entry $\langle g(D), D\rangle$ for this post is enheaped (line 30).

RCA-based Implementation The underlying index structure determines how relevant posts are grouped together. In the inverted index-based RCA approach, posts are spatio-temporally clustered based on their Z-order value. Therefore, a group contains relevant posts that have the same Z-order value.

I^3-based Implementation In the I^3 index, posts are grouped together in octree spatio-temporal cells. Therefore, a group contains relevant posts that reside in the same octree cell.

4.5 Experimental Evaluation

In this section, we present an experimental evaluation of our approach, using two large-scale, real-world datasets of geotagged tweets and photos. We first discuss the experimental setup, outlining the datasets, queries, and parameters used in the experiments, and then we present the results.

4.5.1 Datasets

Next, we describe the two datasets used in the experiments. The first dataset is a collection of geotagged tweets that has also been used in [34] and is provided by the

Table 4.2 Queries used in the experiments.

Query	Term 1	Term 2	Term 3
Q_1	obama	election	president
Q_2	olympic	games	london
Q_3	iphone	apple	ipod
Q_4	nascar	race	car
Q_5	kindle	amazon	ebook
Q_6	nba	basketball	sports
Q_7	economy	market	trading
Q_8	war	weapons	violence
Q_9	concert	festival	show
Q_{10}	vacation	summer	trip

authors[1]. It comprises 20M tweets between April and December 2012. The second dataset comprises photos from Flickr and is provided by Yahoo! [153]. From the original data, we collected a subset of 20M geotagged photos with dates between 2010 and 2014. In both datasets, the posts have a worldwide coverage. The number of distinct keywords is approximately 1.8M for Twitter and 1.3M for Flickr, whereas the average number of keywords per post is 5.7 and 8.4, respectively. The detailed characteristics of the datasets are shown in Table 4.1. The table also shows the disk space required to store the raw files, as well as the constructed indexes, both for the I^3-based and the RCA-based implementations. Note that these values refer to the extended versions of those indexes that include also the time dimension. Moreover, to evaluate the scalability of our approach, we additionally sampled five subsets from each dataset, with sizes ranging from 4M to 20M.

4.5.2 Queries and Parameters

To create a set of realistic and meaningful queries for the above datasets, we combined search terms found in trending Twitter topics in 2012[2], as well as popular tags used in Flickr[3]. The goal was to construct queries that reflect exploratory search, having a few hundreds or thousands of results distributed across space and time. Thus, we selected 10 queries, each one having in turn 3 variants, comprising, respectively, 1, 2, or 3 keywords. The queries used are listed in Table 4.2. In the experiments, we assume OR semantics when using more than one keywords in the query, in order to

[1]http://www.ntu.edu.sg/home/gaocong/datacode.htm
[2]https://2012.twitter.com/en/trends.html
[3]https://www.flickr.com/photos/tags/

Table 4.3 Average number of relevant posts.

Dataset	$\|\Psi\| = 1$	$\|\Psi\| = 2$	$\|\Psi\| = 3$
Twitter	2,891	6,461	13,395
Flickr	486	1,021	1,699

Table 4.4 Parameters used in the experiments.

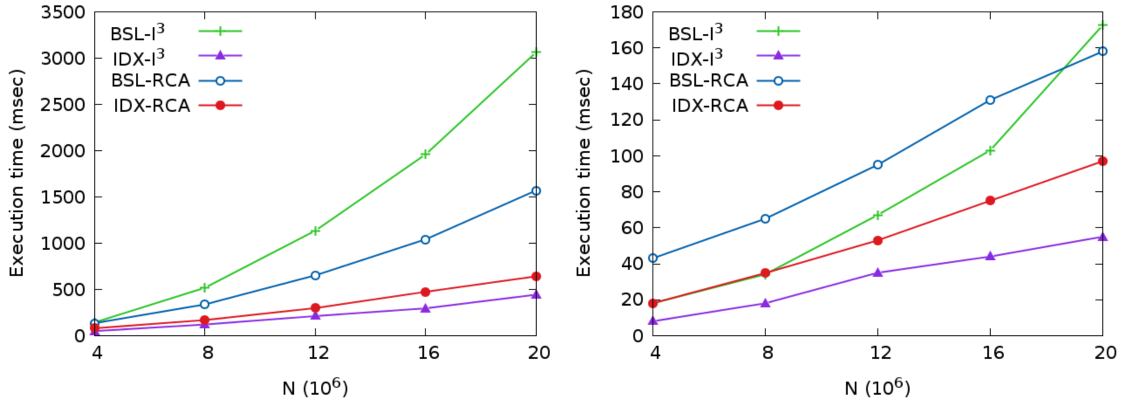
Parameter	**Values**
Number of geotagged posts (N) (10^6)	4, 8, 12, 16, **20**
Number of query keywords ($\|\Psi\|$)	1, **2**, 3
Spatial filter size (R) ($10^6 km^2$)	(approx.) 4, 6, **8**, 10, 12
Temporal filter size (T) (days)	15, 30, **45**, 60, 75
Size of diversified result subset (R^k)	20, 40, **60**, 80, 100
Spatial coverage threshold (ρ_s) (%)	2, 4, **6**, 8, 10
Temporal coverage threshold (ρ_t) (%)	2, 4, **6**, 8, 10

increase the number of relevant posts. Table 4.3 lists the average number of relevant posts for these queries in the Twitter and Flickr datasets (for default values of the spatial and temporal filters R and T).

In addition to query keywords, we also vary the size of the spatial and temporal filters. For the former, we use 5 bounding boxes of increasing sizes over the U.S., covering an area ranging, approximately, from 4 million km^2 up to 12 million km^2. For the latter, we use 5 time intervals starting on 01/08/2012 and having duration from 15 up to 75 days. Moreover, we vary the parameter k, i.e., the size of the diversified result subset, from 20 up to 100. Finally, we experimented with different values for the thresholds ρ_s and ρ_t. These settings are summarized in Table 4.4 (default values are shown in bold).

4.5.3 Dataset Size

Next, we present the results of our experimental evaluation. Specifically, we compare the following four methods: (a) the baseline approach over the I^3-based index (BSL-I^3) and the RCA-based index (BSL-RCA), and (b) our proposed index-aware approach over the I^3-based index (IDX-I^3) and the RCA-based index (IDX-RCA). All algorithms were implemented in Java. In particular, for the I^3 and RCA indexes, we extended the source code that was kindly provided by the authors of [175, 176]. The experiments were conducted on a server with 48GB main memory and Intel® Xeon®



(a) Twitter (b) Flickr

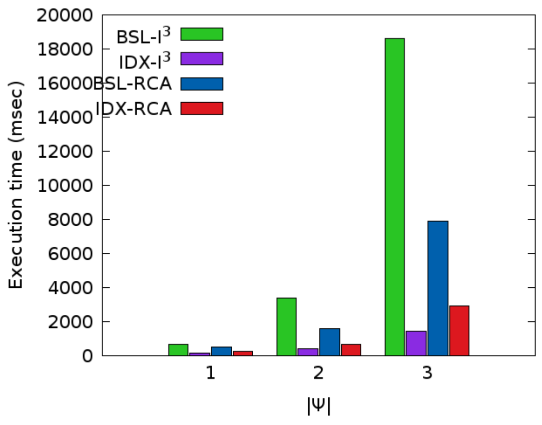
Fig. 4.2 Execution time vs. dataset size.

E5-2420 v2 processor, running Ubuntu 14.04. In each experiment, we vary one of the parameters listed in Table 4.4, setting the rest to their default values. The execution time is then measured by executing each of the 10 queries listed in Table 4.2 5 times and reporting the average.

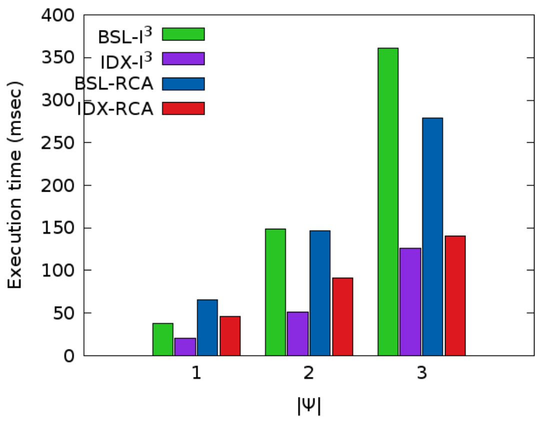
First, we evaluate the scalability of our approach by gradually increasing the dataset size. For this purpose, we have sampled both datasets, Twitter and Flickr, creating five subsets for each, with sizes varying from 4M to 20M posts. The results for the average query execution time are shown in Figure 4.2.

For all methods, execution time increases with the size of the dataset. However, the index-aware approach shows much better scalability compared to the baseline. This observation is particularly evident for the Twitter dataset, while less so for the Flickr dataset. The reason for this has to do with the different selectivity of the queries in the two datasets (see also the discussion in Section 4.5.4). Focusing, for example, on the I^3-based implementation for Twitter, we can observe the following. Although the average query latency for `BSL-I`^{3} starts at below 0.5 seconds, it quickly increases reaching up to more than 3 seconds, whereas at the same time `IDX-I`^{3} still remains below 0.5 seconds. This better scalability of the index-aware method results from the fact that it exploits the underlying index structure to effectively prune large portions of the posts that do not contribute to the final result.

Another interesting observation from the Twitter dataset comes from examining the relative performance of the two different implementations. First, comparing the two baselines, we can see that `BSL-RCA` performs better than `BSL-I`^{3}. Since the baseline method does not exploit the underlying index, this can be attributed to



(a) Twitter



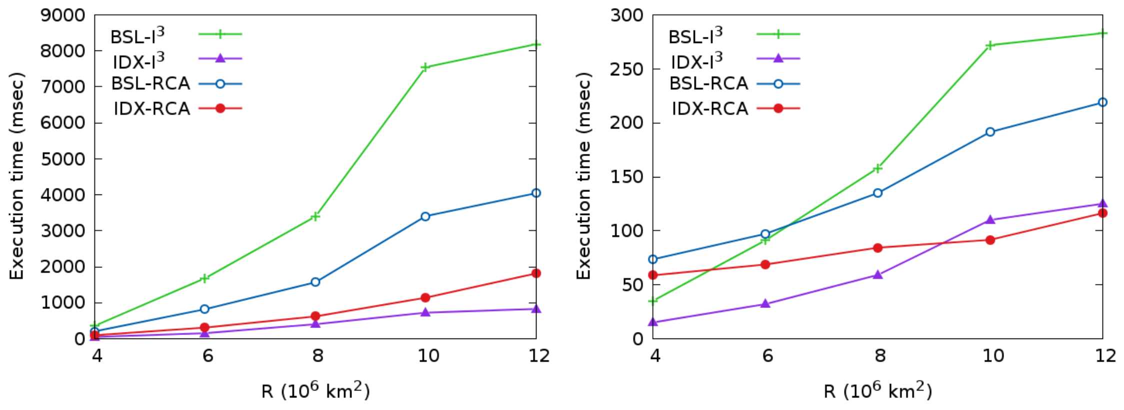
(b) Flickr

Fig. 4.3 Execution time vs. number of keywords.

the fact that the STK filter is evaluated faster with the RCA-based index. However, for the index-aware method, we can see that although both IDX-I^3 and IDX-RCA clearly outperform their respective baselines, the difference is even higher for IDX-I^3, which appears eventually to be slightly faster than IDX-RCA. This indicates that the index-aware method is able to effectively exploit the underlying index in both cases, but the gain is even higher for the I^3-based index. This can be attributed to the fact that the I^3-based index is more effective during spatio-temporal filtering, whereas the RCA-based index, relying on the Z-order encoding, has the additional overhead of filtering out the false positives. This behavior appears to be consistent also for the rest of the experiments described below.

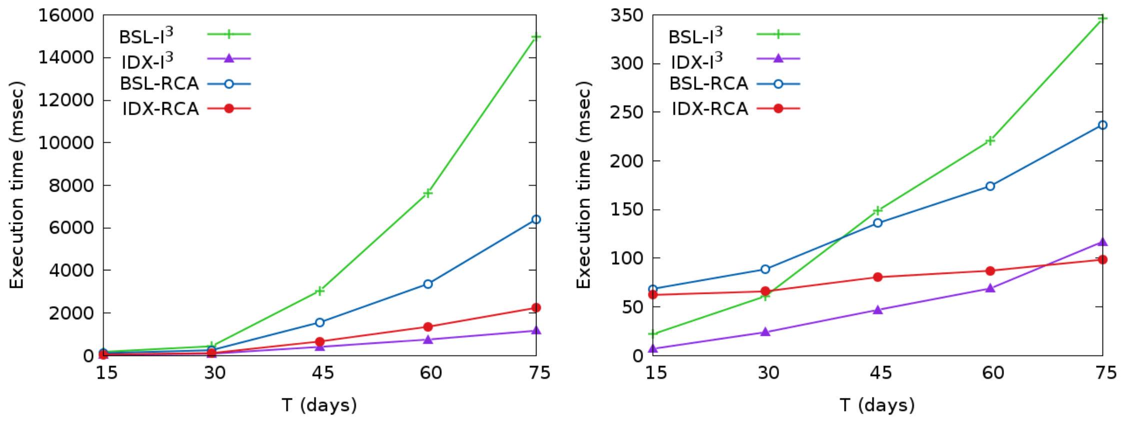
4.5.4 Selectivity of the Conditions in the Query

Next, we examine the effect of changing the selectivity of the query filters. This involves three subsets of experiments, corresponding to each of the dimensions addressed: (a) increasing the number of keywords, (b) increasing the size of the spatial region, and (c) increasing the size of the time window. Each of these conditions is examined separately, and the results are shown in Figures 4.3, 4.4, and 4.5, respectively. Note that increasing the number of keywords (under OR semantics), as well as the size of the spatial or the temporal filter, essentially have the same main effect: the number of relevant posts that match with the STK filter of the query increases. In other words, this increases the size of the original result set, from which the top-*k* results have to be selected.



(a) Twitter (b) Flickr

Fig. 4.4 Execution time vs. spatial region size.



(a) Twitter (b) Flickr

Fig. 4.5 Execution time vs. time window size.

In all experiments, the index-aware methods clearly outperform their respective baselines. More specifically, when the selectivity of the filters is high, the differences are smaller, since the baseline method achieves comparable performance, having to deal with relatively few relevant posts. However, this drastically changes as soon as the filters start to become less selective, allowing for more posts to match. For example, consider the case of the Twitter dataset. Although the average query latency for BSL-I^3 is initially below 1 second, it quickly increases up to 10 seconds or more as the selectivity of the filters decreases. In contrast, IDX-I^3 is significantly less affected, with the average query latency in this case remaining within 1 or 2 seconds, even when the filters reach up to 3 keywords, 12 million km^2, or 75 days. Similar observations can be made also for the Flickr dataset. In that case, although the absolute values of query latency are overall lower, the same differences and trends can be clearly observed. This behavior demonstrates the effectiveness of the pruning strategies and, in particular, the benefit of using the underlying index structure to prune a large number of comparisons when the size of the original set of relevant posts becomes higher.

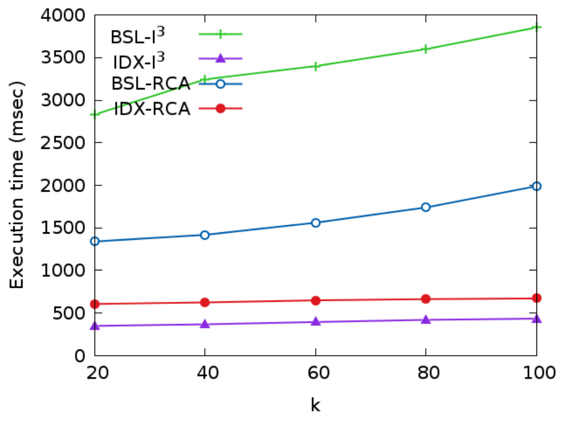
4.5.5 Number of Results

For the next experiment, we evaluate the effect of the parameter k. The results are shown in Figure 4.6. For all cases, the index-aware methods achieve significant gains over their respective baselines. For instance, for the Twitter dataset, the average query latency for `IDX-I`^{3} and `IDX-RCA` remains below 1 second, while reaching up to 4 seconds for `BSL-I`^{3}. The differences are similar for the Flickr dataset as well, with the baseline methods exhibiting even worse scalability in this case. Interestingly, the performance of the index-aware method appears to not be significantly affected by the increase of k. This can be attributed to the fact that, as mentioned in Section 4.4.2, during the iterations that select the next result to be included in the top-k set, some computed values can be cached and reused in subsequent iterations. Thus, although increasing k means that more iterations have to be performed, the additional cost that is incurred gradually decreases.

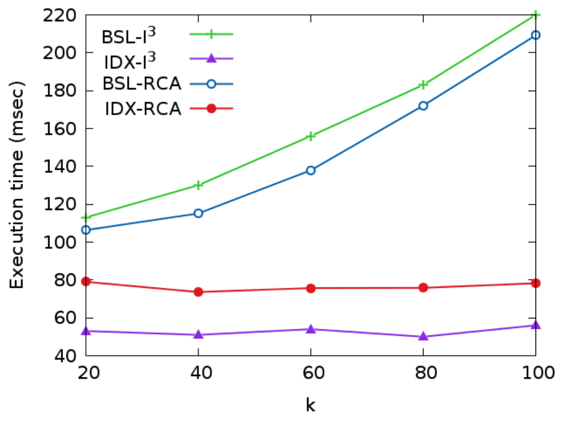
Regarding the comparison between the I^3-based and RCA-based implementations, here we can clearly observe a similar behavior as discussed in Section 4.5.3. For the baseline method, the RCA-based index again performs better, requiring less time to apply the STK filter. However, this difference is eventually overcome as the index-aware method is again able to more effectively exploit the I^3-based index. Thus, the final result is reversed, with `IDX-I`^{3} achieving the best performance.

4.5.6 Coverage Thresholds

Finally, we examine the effect of the spatial and temporal thresholds, ρ_s and ρ_t, which determine the radius for coverage for each document. The results are shown in Figure 4.7. Again, query latency is significantly lower for the index-aware methods compared to the baselines. In addition, we can see that the baseline methods do not seem to be affected by this parameter, since the number of comparisons that need to be performed is not affected by these values. Interestingly, this can also be observed for the index-aware methods. Notice that these thresholds can be set at query time,

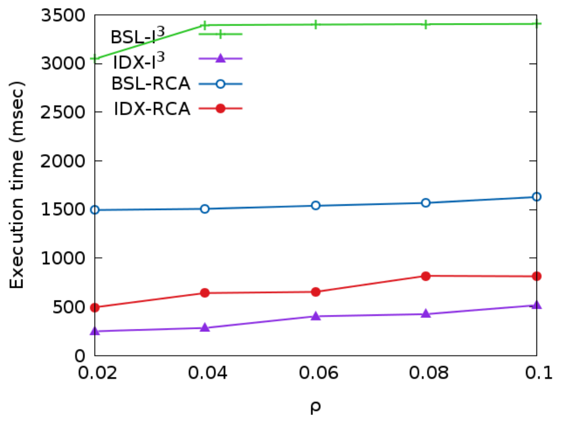


(a) Twitter

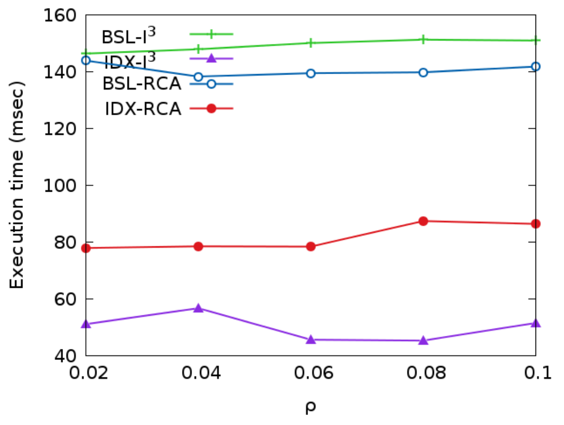


(b) Flickr

Fig. 4.6 Execution time vs. number of results.



(a) Twitter



(b) Flickr

Fig. 4.7 Execution time vs. coverage thresholds.

thus the underlying index structure is constructed independently of them. Hence, this observation shows that the proposed approach is robust, in the sense that it does not require to fine tune the underlying index according to these thresholds in order to achieve a benefit through the pruning.

Moreover, comparing the performance of the I^3-based and RCA-based implementations, the results are consistent with the findings of the previous experiments. Again, `BSL-RCA` shows an advantage over `BSL-I`^{3}, but `IDX-I`^{3} achieves the best performance, having a higher gain that overcomes also this initial difference.

4.6 Summary

In this chapter, we have introduced a novel type of spatial-temporal-keyword query, the *k*`CD-STK` query. This query is based on two key notions, *spatio-temporal coverage* and *diversity*, which are formally defined. In particular, the query is formulated similarly to other search results diversification problems, which allows us to derive a baseline approach for its evaluation. Then, we focus on developing a more efficient strategy for processing *k*`CD-STK` qeuries, which allows to exploit an underlying hybrid (spatial-temporal-keyword) index not only for the first part, i.e., the filtering of relevant posts, but also for the second part, i.e., the selection of the top-*k* results based on coverage and diversity. To that end, we have considered two state-of-the-art spatio-textual indexes, which we extended to include also the time dimension, and we have shown how our proposed index-aware approach can be applied on top of those structures.

To validate and evaluate our approach, we have conducted an experimental evaluation on large real-world datasets containing geotagged tweets and photos. The results have shown that our optimized approach manages to successfully exploit the available index to significantly reduce the query execution time compared to the baseline algorithm. This holds for both indexes that have been considered, namely the I^3-based and the RCA-based implementations.

Chapter 5

Continuous Summarization of Streams of Posts

We have already explained our method for finding a set of representative results for the spatio-temporal exploration of a large number of posts in the preceding chapter. However, there the result set was computed in an ad hoc manner and did not change with time. Here, we address this limitation and examine the problem of continuous spatio-textual summarization of streams.

5.1 Overview

As discussed previously, the spatio-textual data available in geotagged posts and other online sources provides a valuable source for analysis and mining, e.g., for identifying and monitoring events at various locations, mining trending topics in different areas, studying the spatial distribution of opinions and sentiments associated with various entities, etc. However, given the high rate at which this content is constantly produced, it easily becomes difficult and overwhelming for the user to keep track of the whole stream of information. Moreover, this stream is typically characterized by a high degree of repetition and redundancy, e.g., same or similar news articles and stories being published by several sources, same information being re-tweeted by multiple users, similar opinions and comments being expressed, etc. Thus, for many applications and needs, it is often impractical or even uninteresting to actually keep track of the whole stream of posts. Instead, it is sufficient or desirable to compute and maintain a more concise, aggregate *summary* of relatively few, representative posts. Consequently, it has become an important task for many search engines, recommendation engines, and publish/subscribe systems to maintain a small, diversified set of posts over this streaming content, in order to provide to the users a summarized overview of the underlying content and anchor points for further exploration.

Diversifying search results or, more generally, a subset of documents selected from a larger collection, is already an established and well-studied problem in the fields of information retrieval and web search, and it has been shown to improve the quality of produced results and user satisfaction in several practical applications [73, 50, 158]. The diversity of a selected set of documents is a measure of the dissimilarity of these documents to each other. Increasing diversity essentially increases the variety of contents. Thus, given a document collection that is characterized by a high degree of overlap and repetition, or a query that is inherently ambiguous or opinionated, favoring the selection of a document summary that is more diverse can increase the coverage of different topics, aspects, opinions, or sentiments, thus reducing bias.

To that end, the document selection algorithm takes into consideration the criterion of *diversity* of the selected documents to each other, in addition to the standard criterion of *coverage* of each individual document to the query (or, more generally, any other measure of individual importance of each document). Several diversification models and formalisms have been proposed in the literature. Since the problem of finding the optimal diversified subset of documents under various formulations is NP-hard, various heuristics are employed to compute approximate solutions more efficiently [73].

However, most of the existing approaches only address *static* settings (i.e., computing a diversified subset of a given document collection or of the result set of a given query), whereas very few ones have considered a dynamic or *streaming* context [51, 126]. Moreover, those that do, typically employ various restrictions and assumptions that simplify the problem, making it easier to tackle, but at the expense of restricting flexibility and generality. Furthermore, in both static and dynamic/streaming contexts, the exact type of contents of the handled documents, and respectively the exact type of coverage/relevance scores and dissimilarity measures defined on them, is considered as an orthogonal issue. Thus, any proposed optimizations consider only the generic diversification algorithm itself, without further optimizing the process based on the kind of documents handled.

In this chapter, we attempt to fill these gaps, focusing on computing and maintaining spatially and textually diversified summaries over a stream of spatio-textual posts. Specifically, we adopt the *sliding window* model by examining successive chunks of the incoming stream and incrementally updating the resulting summary to reflect recently trending posts. First, we present different diversification strategies that provide different trade-offs between maximizing the quality (i.e., combined coverage and diversity) of the resulting subset and minimizing computational cost. Then, we turn our focus to the specific case of spatio-textual posts, proposing optimizations that can be applied to enhance the efficiency of those diversification strategies in this class of content. To the best of our knowledge, our work is the first one to address the problem of maintaining a diversified summary of posts over a stream of spatio-textual documents.

Our main contributions in this chapter can be summarized as follows:

- We formally define the problem of summarizing a stream of spatio-textual posts over a sliding window, defining specific spatio-textual criteria of coverage and diversity.
- We propose different stream summarization strategies that provide a trade-off between result quality and computational cost.
- We optimize our proposed strategies through the use of lightweight spatio-textual structures maintained over the sliding window which are used to update the result set efficiently at every step.
- We present an experimental evaluation to study and compare the performance trade-offs of our proposed methods and the resulting performance gains of our algorithms.

The remainder of this chapter is organized as follows. Section 5.2 reviews related work on summarization and diversification methods. Section 5.3 formally defines the problem and the criteria for spatio-textual coverage and diversity. Then Section 5.4 presents our solution to the continuous summarization problem, while Section 5.5 discusses optimizations based on spatio-textual partitioning. Section 5.6 presents an experimental evaluation of our methods, and Section 5.7 concludes the chapter.

5.2 Additional Relevant Background

As we discuss later, our problem formulation in the non-dynamic case is an instance of the max-sum diversification problem [73]. Therefore, we first present an overview of document summarization focusing on diversification-based techniques, and then discuss the case of diversification over streaming data.

5.2.1 Summarization via Diversification

The seminal work of [25] studied the problem of document summarization and formulated it as an instance of the diversification problem, which was later studied in various incarnations. Diversification in general aims at reducing repetition and redundancy in the result set returned to the user by selecting the final set of results based not only on each document's relevance to the query, but also on the dissimilarity of the selected results to each other. This increases the variety and novelty of the information included in the diversified result set, which is particularly important for queries that are inherently ambiguous or for which there exist different subtopics, perspectives, opinions, and sentiments. In such cases, diversification allows to better cover and represent these different aspects in the result set.

Many different formulations can be used to formally define the objective of search results diversification (see [73, 50, 158] for an overview and classification of existing approaches). Perhaps the most well-known approach is the framework proposed in [73], which was also discussed in Chapter 4. We give a brief overview again here for convenience. According to [73], given a query Q and an initial result set R containing all documents relevant to Q, the goal is to select a relatively small subset R^* of R, with $|R^*| = k$, that maximizes an objective function ϕ. The latter combines: (a) a *relevance score*, assessing how relevant each document in R^* is to the given query, and (b) a *diversity score*, measuring how diverse the documents in R^* are to each other. Function ϕ can take different forms, such as *max-sum*, *max-min*, and *mono-objective* diversification. For instance, in the *max-sum* variant, ϕ is defined as the weighted sum of (a) the total relevance of the documents in R^* to Q, and (b) the sum of pairwise distances among the documents in R^*.

Typically, finding the exact result set that maximizes the diversification objective is NP-hard. Thus, approximate solutions are proposed by each method. These usually rely either on *greedy* heuristics, which build the diversified set incrementally, or on *interchange* heuristics, which gradually improve upon a randomly selected initial set by swapping its elements with other ones that improve its diversity. In a recent work [27], the notion of approximate, composable core-sets has been used to address the k-diversity maximization problem in general metric spaces. A core-set is a small set of items that approximates some property of a larger set [90]. Based on these, the authors develop efficient algorithms for diversity maximization in the streaming and MapReduce models.

Moreover, summarization has also been studied in the context of coverage alone, without a notion of diversity [150, 108]. Such a related coverage problem is presented in [52]. In this case, the goal is to select the minimum subset of documents that are diverse to each other, i.e., have distance to each other at least ϵ, and cover the whole dataset, i.e., each non-selected document lies within distance at most ϵ from a selected one. However, the size of the summary is not fixed, but rather depends on the distance threshold ϵ.

5.2.2 Diversification over Streaming Data

Few works so far have considered the problem of continuously maintaining a diversified result set over streaming data. In [51], the problem of continuous diversification over dynamic data is considered. The proposed approach adopts the *max-min* objective function for diversification, which entails maximizing the minimum distance between any pair of documents in the result set. The proposed method relies on the use of *cover trees* and provides solutions with varying accuracy and complexity for selecting items that are both relevant and diverse. Moreover, it introduces a sliding window model for coping with the continuous variant of the problem against streaming items. However, a cover tree needs to be incrementally updated by keeping all raw items within the current window, which can have a prohibitive cost in space and time when dealing with massive, frequently updated streaming data. In our case, we rely on much more lightweight aggregate information to speed up the computation of the new result set every time the window slides.

The work presented in [126] assumes a *landmark window* model, i.e., a window over the stream that spans from a fixed point in the past until the present. Based on this, an online algorithm is proposed, which checks every new incoming document against those in the current result set and performs a substitution if it increases the objective score of the set. Thus, this method addresses the diversification problem on an ever increasing stream of objects. However, it is restricted by the fact that it always considers exactly one incoming document and does not handle document expiration. Section 5.4.2 revisits this approach and describes an adaptation of their algorithm to our problem involving sliding windows, i.e., where both the start and the end points of the window slide.

5.3 Model and Definitions

Spatio-Textual Stream. We consider geotagged messages posted by users of social networks or microblogs (e.g., Facebook, Twitter, Flickr, Foursquare) in a streaming fashion. We assume that each post p comprises textual information and a geolocation, as defined next.

Definition 5.1 (Post). *A spatio-textual post $p = \langle \Psi, \ell, t \rangle$ consists of a collection of keywords Ψ from a vocabulary V and was generated at location ℓ (specified as a pair of coordinates (x, y)) at timestamp t.*

We assume that all posts are available as a stream of tuples. We adopt a time-based *sliding window* model [105] over the stream, as defined below.

Definition 5.2 (Sliding Window). *A time-based sliding window $\mathcal{W}$ is specified with two parameters: (a) a* range *spanning over the most recent ω timestamps backwards from current time t_c, and (b) a* slide step *of β timestamps. Upon each slide, window $\mathcal{W}$ moves forward and provides all messages posted during time interval $(t_c - \omega, t_c]$. These messages comprise the current* state *of the window, i.e.,*

$$\mathcal{W} = \{p : p.t \in (t_c - \omega, t_c]\}. \tag{5.1}$$

Posts with timestamps earlier than $t_c - \omega$ are called expired.

Spatio-Textual Summary. Our goal is to select an appropriate subset of the posts in the window to form an informative summary. A summary S of window $\mathcal{W}$ is a subset of the posts in $\mathcal{W}$.

In accordance with work on document summarization [72, 108], given a constraint on the maximum summary size, our objective is to construct a summary that *covers* as much as possible the entire set of posts in the current window while at the same time containing *diverse* information as much as possible. Formally, we capture these two requirements using the two measures defined next.

Definition 5.3 (Coverage). *The* coverage *$cov(S)$ of a summary S captures the degree to which the posts in the summary approximate the spatial and textual information in the window. We use a weight α to capture the relative importance of the two information facets:*

$$cov(S) = \alpha \cdot cov^T(S) + (1 - \alpha) \cdot cov^S(S). \tag{5.2}$$

Similar to [108], we define the *textual coverage* of a summary as

$$cov^T(S) = \sum_{p_i \in \mathcal{W}} \sum_{p_j \in S} sim^T(p_i, p_j), \tag{5.3}$$

where $sim^T(\cdot, \cdot)$ is a textual similarity metric between posts.

For our purposes, we consider the vector space model, and define $sim(\cdot,\cdot)$ as the *cosine similarity* of the vector representations of the posts. Specifically, each space coordinate corresponds to a keyword, and the vector's coordinate contains a weight representing the importance of the corresponding keyword relative to the window. While any tf-idf weighing scheme [142, 119] is possible, here we simply use term frequency and normalize the vectors to unit norm. Therefore, the textual coverage is computed as the sum over each pair of posts (one from the window and another from the summary) of the inner product of their vector representations:

$$cov^T(S) = \sum_{p_i \in \mathcal{W}} \sum_{p_j \in S} \sum_{\psi} p_i[\psi] \cdot p_j[\psi], \tag{5.4}$$

where keyword ψ is used to index the vector, and thus $p[\psi]$ denotes the normalized weight of keyword ψ of post p.

For the *spatial coverage*, we follow a similar formulation and define it as cosine similarity in a (different) vector space. Instead of keywords from a vocabulary, we have a set of regions from a predetermined spatial partitioning ρ (e.g., regions could represent cells of a uniform grid). Intuitively, such a coarse partitioning allows for a macroscopic view of the posts in the window, where exact post locations are not important, and thus coalesced into broader regions. As each post is always associated with a single region, the spatial content of a post is simply represented as a vector having a single weight 1 at the vector coordinate representing the region containing the post's geotag. Thus, the spatial coverage is computed as:

$$cov^S(S) = \sum_{p_i \in \mathcal{W}} \sum_{p_j \in S} \left|\rho(p_i.\ell) = \rho(p_j.\ell)\right|, \tag{5.5}$$

where $\rho(\ell)$ is the region associated with location ℓ, and $|\rho(p_i.\ell) = \rho(p_j.\ell)|$ returns 1 if locations $p_i.\ell$, $p_j.\ell$ reside in the same region.

Next, we define the *diversity* of a summary.

Definition 5.4 (Diversity)**.** *The* diversity $div(S)$ *of a summary S captures the degree to which the posts in S carry dissimilar information. As before, diversity is defined as the weighted sum of a textual and spatial term:*

$$div(S) = \alpha \cdot div^T(S) + (1-\alpha) \cdot div^S(S). \tag{5.6}$$

Textual diversity is defined with respect to the vector space model. Specifically, textual diversity is the sum of cosine distance between all pairs of posts in the summary:

$$div^T(S) = \sum_{\{p,p'\}:p \neq p' \in S} \left(1 - \sum_{\psi} p_i[\psi] \cdot p_j[\psi]\right). \tag{5.7}$$

On the other hand, *spatial diversity* is defined based on a spatial distance (e.g., Euclidean, haversine) between summary posts' exact locations:

$$div^S(S) = \sum_{\{p,p'\}:p \neq p' \in S} dist(p.\ell, p'.\ell). \tag{5.8}$$

Based on the definitions of these two quality measures of a summary, we are now ready to state our problem.

Definition 5.5 (Stream Summarization). *For each sliding window $\mathcal{W}$ over a stream of posts, determine the summary S^* of size k that maximizes the objective function:*

$$S^* = \underset{S \subseteq \mathcal{W}, |S|=k}{\arg\max}\ f(S),$$
$$f(S) = \lambda \cdot cov(S) + (1-\lambda) \cdot div(S),$$

where λ determines the trade-off between coverage and diversity.

5.4 Algorithmic Approach

If we consider any individual instantiation of the sliding window, our problem formulation is identical to the max-sum diversification problem. Thus, one can apply the adaptation of the greedy algorithm in [15] to summarize the contents of each window. There, the authors observed that the simple greedy heuristic from [137] can give a linear time 2-approximation. We call this algorithm GA — introduced as 1-Greedy Augment in [15]. GA starts with a random object in the result set and iteratively appends to the result the object maximizing the marginal gain. However, such an approach is impractical simply because the sliding window can be arbitrarily large and storing its entire contents is not an option. Therefore, we need to devise an efficient solution that operates on limited memory.

To achieve this we need to address two tasks. The first is *how to compute the coverage* of posts without having the window's contents. Recall that the coverage of a single post is computed as the sum of its cosine similarity with each post in the window. The second task is *how to construct the summary* without having the window's contents. While this problem has been studied for landmark windows with limited memory [126] and sliding windows without memory restrictions [51], to the best of our knowledge it has not been addressed for sliding windows under limited memory. Section 5.4.1 addresses the first task, while Section 5.4.2 discusses the second.

5.4.1 Computing Coverage

To compute the coverage without keeping the entire window contents, we exploit the linearity of the inner product — the cosine similarity of two normalized vectors is their inner product. Note that in the following discussion, we use the term coverage to refer to both textual and spatial coverage, as they are both defined as a sum of inner products.

Our approach is based on the notion of window *pane* (or sub-window) [105]. For ease of presentation, we assume that the size of the window ω is a factor of its slide step β, e.g., a window of 24 hours sliding every one hour. The window is thus naturally divided into $m = \omega/\beta$ panes. Each time the window slides, all tuples within the oldest pane expire, while new tuples arrive in the newest pane, termed *current*.

In what follows, we denote as $\mathcal{W}$ the current and as $\mathcal{W}'$ the previous window instantiation. We also denote as $\mathcal{W}^-$ the expired pane of the previous window and refer to the current pane as $\mathcal{W}^+$, i.e., $\mathcal{W}^- = \mathcal{W}' \setminus \mathcal{W}$ and $\mathcal{W}^+ = \mathcal{W} \setminus \mathcal{W}'$. When we want to enumerate the panes of the window we simply use the notation $\mathcal{W}_1$ through $\mathcal{W}_m$.

For each pane $\mathcal{W}_i$, we define its information content W_i as the vector:

$$W_i = \sum_{p \in \mathcal{W}_i} p. \tag{5.9}$$

It is then easy to see that the coverage of a post p can be efficiently computed using the information contents of the m panes:

$$cov(p) = \sum_{i=1}^{m} \sum_{\tau} W_i[\tau] \cdot p[\tau], \tag{5.10}$$

where τ represents either a keyword or a region. This implies a simple solution to compute the coverage. Instead of requiring the set of all posts within a window, it suffices to store only a few vectors that are the information content of each pane. When the window slides, we just throw away the information content of the expired pane and begin aggregating posts in the current pane to form its information content.

5.4.2 Building the Summary

In this section, we describe several strategies for building a summary over the sliding window of posts. All approaches (except the baseline) operate without storing the entire contents of the window. They differ in what (limited) information they store across the window panes and in the way they construct the summary or update the previous one. In all strategies, we describe the operation necessary in a single window. We assume that the information content for the current pane has been constructed as per the previous section, and thus the coverage of any post can be computed using Equation 5.10.

Baseline Strategy

The baseline strategy (BL) requires storing the entire contents of the window and is thus impractical, serving only as a benchmark for the quality of the summary. It builds the summary incrementally, starting with an empty set. Then, at each step it inserts the post that maximizes the marginal gain of the objective function. Given a summary S, the marginal gain of a post p is:

$$\phi(p) = \lambda \cdot cov(p) + (1 - \lambda) \cdot div(p, S). \tag{5.11}$$

Note that GA initializes the summary with a random object because it cannot differentiate among objects when the summary is empty. On the other hand, BL can differentiate among posts, and thus it selects as the first post the one that has the largest coverage.

If we assume m panes in the window, each with an equal number n of posts, we see that the memory footprint of BL is $O(k + m \cdot\ n)$. BL performs k passes over all posts, computing for each post its diversity with respect to at most k summary posts. Thus, BL requires time $O(k^2 \cdot m \cdot n)$ to construct the summary of a window. Its running time can be improved in practice using the techniques described in Section 5.5.

Online Interchange Strategy

The work in [126] describes an online algorithm for solving the max-sum diversification problem on an ever increasing stream of objects. This approach essentially solves the problem for a landmark window, which spans from a fixed point in the past until the present. For our purposes, we adapt this algorithm to our problem involving

Algorithm 5.1: Online Interchange

```
Input:
Output:
1 $S \leftarrow S' \setminus \mathcal{W}^-$
2 foreach $p \in \mathcal{W}^+$ do                    ▷ examine new posts in chronological order
3     if $|S| < k$ then
4         insert $p$ into $S$
5     else
6         $p^- \leftarrow \arg\max_{p' \in S} f(S \setminus \{p'\} \cup \{p\})$
7         if $f(S \setminus \{p^-\} \cup \{p\}) > f(S)$ then
8             $S \leftarrow S \setminus \{p^-\} \cup \{p\}$                    ▷ replace $p^-$ with $p$
9 return $S$
```

sliding windows, where both the start and the end point of the window slide. We refer to this algorithm as OI.

Algorithm 5.1 presents the pseudocode for constructing the summary S of the current window by making incremental changes to the summary S' constructed for the previous window. Initially, the summary is constructed as the previous summary excluding any expired posts contained in that summary (line 1). Then, each newly arrived post p is examined in sequence (lines 2–8). If the summary is not yet full, the post is simply inserted (lines 3–4). Otherwise, the algorithm identifies the best post p^- to evict from the summary in favor of the current examined post p (line 6). If the eviction of p^- and the insertion of p results in an increase of the objective function, the algorithm proceeds with the replacement (lines 7–8).

The OI algorithm operates on limited memory, requiring space of $O(k+n)$. For each post in the current pane, OI computes the objective score of k possible sets (one for each possible substitution of the post in the summary). Because these examined sets have significant overlap with each other, the computation of the objective score for each set can be efficiently implemented in $O(k)$ time by some clever bookkeeping [126]. Thus, the running time of OI is $O(k^2 \cdot n)$. Unfortunately, OI cannot take advantage of the optimization discussed in Section 5.5.

Oblivious Summarization

The oblivious summarization (OS) strategy, in contrast to OI, does not try to improve on the existing summary. Rather, it rebuilds the summary from scratch selecting among the posts in the current pane and those (not expired) in the previous summary. Therefore, it applies the GA algorithm on the set $(S \setminus \mathcal{W}^-) \cup \mathcal{W}^+$. Naturally, the difference with BL is in the posts considered for inclusion in the summary; BL considers all window posts, whereas OS has fewer options.

The OS strategy requires space $O(k+n)$ and makes k passes over all n posts in the current pane. For each post, it computes its diversity with respect to at most k summary posts. Thus, the running time of OS is $O(k^2 \cdot n)$. The OS strategy can also benefit from the optimization of Section 5.5.

Intra-Pane Summarization

The key idea of intra-pane (IP) summarization is to store a brief summary over each pane, and then use these summaries to derive a summary for current window. Therefore, at each window slide, IP constructs a local summary of size k' of the current pane using the GA algorithm. This summary is then stored along the pane unaltered until its expiration. To compute the window summary, IP once again employs the GA algorithm, but this time over the summary posts of each pane.

IP requires k' space for each pane, in addition to storing the contents of the current pane. Therefore, it requires space $O(k' \cdot m)$. For a given window, IP invokes GA two times, once to construct the current pane's local summary with a running time of $O(k'^2 \cdot n)$, and another to construct the window summary over the pane summaries (a total of $k' \cdot m$ posts) with a cost of $O(k^2 \cdot k' \cdot m)$.

5.5 Spatio-Textual Optimizations

The main bottleneck in all methods described above is that each post has to be evaluated individually regarding its suitability for being included in the summary. However, in practice, many posts may be similar to each other. In that case, it is possible to group such similar posts together and then make a decision collectively for the group, i.e., whether any post among those should be included in the summary. In the following, we elaborate on this idea and present a process for achieving this purpose.

The process comprises two stages. The first involves partitioning the available posts into groups. Then, given a group of posts and a summary, the second is to establish upper and lower bounds for the coverage and the diversity of each post in the group with respect to the given summary. Next, we describe our method for partitioning the posts, and then we present how the bounds are computed.

5.5.1 Spatio-Textual Partitioning

Partitioning the posts in each pane is based on both their spatial and textual information. Given that each post belongs to exactly one region ρ, we adopt a *spatial-first* partitioning, e.g., a uniform grid partitioning into cells or a planar tessellation into non-overlapping tiles [111]. Let $\mathcal{P}$ denote a set of posts contained within the same spatial partition. Then, the next step is to further partition $\mathcal{P}$ textually, so that the resulting subsets of posts are as homogeneous as possible with respect to the keywords they contain. The latter condition is helpful for deriving tighter bounds. Based on this, we formulate next the criterion for the textual partitioning.

Let $\Psi_{\mathcal{P}}$ denote the union of the keyword sets of the posts in $\mathcal{P}$. Assume also a partitioning Γ of $\mathcal{P}$ into the subsets $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{|\Gamma|}$. We define the *gain* $g(\mathcal{P}, \mathcal{P}_i)$ of each subset $\mathcal{P}_i$ w.r.t. $\mathcal{P}$ as the reduction rate of the size of the corresponding keyword set, i.e.,

$$g(\mathcal{P}, \mathcal{P}_i) = \frac{|\Psi_{\mathcal{P}}| - |\Psi_{\mathcal{P}_i}|}{|\Psi_{\mathcal{P}}|}. \tag{5.12}$$

This implies that the gain is higher for partitions that have a lower number of distinct keywords. Then, the overall gain resulting from partitioning Γ is defined as:

$$g(\mathcal{P}, \Gamma) = \frac{\sum_{\mathcal{P}_i \in \Gamma} g(\mathcal{P}, \mathcal{P}_i)}{|\Gamma|}. \tag{5.13}$$

Using this gain function, we can partition the initial set of posts recursively, applying a greedy algorithm. At each iteration, the algorithm selects one keyword for splitting and partitions the initial set into two subsets, according to whether each post contains that keyword or not. Selecting the keyword on which to split is based on finding the keyword which results in the partitioning with the maximum gain. Then, each of the resulting subsets is partitioned recursively, until the desired number of partitions is reached or until there is no significant gain by further partitioning.

Nevertheless, performing the above check over all the candidate keywords during each iteration is time consuming. A compromise is to perform this computation offline, at a lower rate, or using a subset of the stream, to identify a set of keywords that are good candidates for partitioning, and then apply these ones, updated periodically, to partition the posts in each pane. An even simpler alternative is to rely on the most frequent keywords for partitioning, since the keyword frequencies are already known for each previous pane in the window, thus requiring no additional overhead. Note that regardless of the way the partitioning is done, the correctness of the bounds presented in the next section is not affected.

5.5.2 Coverage and Diversity Bounds

In what follows, we focus on a particular partition $\mathcal{P}$ of our spatio-textual partitioning and describe the necessary aggregate information we need to store and how to derive upper bounds on coverage and diversity. We abuse notation and also denote by $\mathcal{P}$ the set of posts indexed in any sub-partition below the examined.

We associate with $\mathcal{P}$ the following information:

- a vector $\mathcal{P}.p^+$, which stores at each coordinate the highest weight seen among all posts in $\mathcal{P}$;
- a vector $\mathcal{P}.p^-$, which stores at each coordinate the lowest weight seen among all posts in $\mathcal{P}$;
- the set $\mathcal{P}.\Psi$ of all keywords appearing in a post in $\mathcal{P}$.

Using this information, we next discuss how we derive the bounds.

Coverage

We firstly compute an upper bound to the possible textual coverage of a post in $\mathcal{P}$ with respect to the information content W of the window or the current pane. In other words, we seek an upper bound to

$$\max_{p \in \mathcal{P}} \sum_{\psi} W[\psi] \cdot p[\psi]. \tag{5.14}$$

We construct two bounds and select the tighter one. The first trivially uses the $\mathcal{P}.p^+$ vector to upper bound a post from $\mathcal{P}$:

$$cov^T(p \in \mathcal{P})^+_I = \sum_{\psi} W[\psi] \cdot \mathcal{P}.p^+[\psi]. \tag{5.15}$$

The second is based on the property that the cosine of two vectors is maximized when the vectors are parallel to each other. In our case, this translates to constructing a unit vector x parallel to W (unit because vectors in $\mathcal{P}$ are normalized). However, the inner product of W with this x would overestimate the coverage of posts in $\mathcal{P}$. As a matter of fact, vector x is constructed independently of the posts within partition $\mathcal{P}$, and thus such an upper bound trivially applies to all partitions. A tighter bound can be derived if we first project W to the dimensions corresponding to keywords in $\mathcal{P}$, and then take the unit vector parallel to W. Therefore, the second upper bound is:

$$cov^T(p \in \mathcal{P})^+_{II} = \frac{1}{\|W'\|} \cdot \sum_{\psi} W[\psi] \cdot W'[\psi] = \|W'\|, \tag{5.16}$$

where W' is the aforementioned projection of W, i.e.,

$$W'[\psi] = \begin{cases} W[\psi] & \text{if } \psi \in \mathcal{P}.\Upsilon \\ 0 & \text{otherwise.} \end{cases} \tag{5.17}$$

We can now prove the following.

Lemma 5.1. *The previously defined $cov^T(p \in \mathcal{P})^+_I$ and $cov^T(p \in \mathcal{P})^+_{II}$ are upper bounds to the coverage of any post p in partition $\mathcal{P}$.*

Proof. The lemma holds for the first upper bound because for every keyword ψ we have that $p[\psi] \leq \mathcal{P}.p^+[\psi]$.

For the second upper bound, it is easier to work with a vector notation. The maximum coverage of any post p is the maximum inner product of any p with the information content W, i.e., $\max_{p \in \mathcal{P}} W \cdot p$. Because p has zero coordinates at any $\psi \notin \mathcal{P}.\Upsilon$, the previous is equal to the maximum inner product of any p with W', i.e., $\max_{p \in \mathcal{P}} W' \cdot p$. Since p is a unit vector, its maximum inner product with W' cannot be greater than the norm of W'. □

Regarding the spatial coverage, observe that all posts in the partition have the same coverage (they fall in the same region), which is computed exactly as $cov^S(p \in \mathcal{P}) = \sum_{p' \in \mathcal{W}} |\rho(p'.\ell) = \rho(\mathcal{P}.\ell)|$.

Diversity

Next, our goal is to compute an upper bound to the possible textual diversity of a post in $\mathcal{P}$ with respect to summary S, i.e., we seek an upper bound to

$$\max_{p \in \mathcal{P}} \sum_{p' \in S} \left(1 - \sum_{\psi} p'[\psi] \cdot p[\psi]\right) = |S| - \min_{p \in \mathcal{P}} \sum_{p' \in S} \sum_{\psi} p'[\psi] \cdot p[\psi].$$

Similar to the case of coverage, we can derive a diversity upper bound in two ways and employ the tighter. The first is by using the $\mathcal{P}.p^-$ vector to lower bound the inner products:

$$div^T(p \in \mathcal{P}, S)^+_I = |S| - \sum_{p' \in S} \sum_{\psi} p'[\psi] \cdot \mathcal{P}.p^-[\psi]. \tag{5.18}$$

The second is again based on a geometric property of the inner product. In general, the inner product between two vectors is minimized when the vectors are parallel but in opposite directions. In the case of non-negative vectors, given a vector p', the non-negative unit vector x that maximizes their inner product must be parallel to one of the axes (intuitively, in a direction as far away from p' as possible), and in particular the axis where p' has its smallest coordinate. To construct a tighter bound in our setting, we need to consider only the axes (dimensions) corresponding to keywords present in posts of $\mathcal{P}$. Therefore, the second upper bound is:

$$div^T(p \in \mathcal{P}, S)^+_{II} = |S| - \sum_{p' \in S} \min_{\psi \in \mathcal{P}.\Psi} p'[\psi]. \tag{5.19}$$

Lemma 5.2. *The previously defined $div^T(p \in \mathcal{P}, S)^+_I$ and $div^T(p \in \mathcal{P}, S)^+_{II}$ are upper bounds to the diversity of any post p in partition $\mathcal{P}$ to summary S.*

Proof. The proof for the first upper bound follows from $p[\psi] \geq \mathcal{P}.p^-[\psi]$.

For the second bound, we use the following inequality:

$$\min_p \sum_i x[i] \cdot p[i] \geq \min_{i:p[i] \neq 0} x[i],$$

which holds for any unit vector p with positive coordinates.

Applying the inequality for vector $x = \sum_{p' \in S} p'[\psi]$ we get that $\sum_{p' \in S} p'[\psi] \geq \min_{\psi \in \mathcal{P}.\Psi} p'[\psi]$, where condition $\psi \in \mathcal{P}.\Psi$ is equivalent to $\psi : p[\psi] \neq 0$. The lemma follows after multiplying the resulting inequality with -1 and adding $|S|$. □

Regarding spatial diversity, we can upper bound it using the maximum possible distance between summary posts and the minimum bounding rectangle (MBR) of all posts in $\mathcal{P}$.

$$div^S(p \in \mathcal{P}, S)^+ = \sum_{p' \in S} \text{maxdist}(p', \mathcal{P}), \tag{5.20}$$

where $\text{maxdist}(p', \mathcal{P})$ returns the maximum distance of the $\mathcal{P}$'s MBR to point p'.

5.6 Experimental Evaluation

In this section, we describe our experimental setting and present the results of our experiments. We start by describing the datasets used in the experiments, the parameters involved, and the performance criteria used to evaluate the various methods.

5.6.1 Datasets

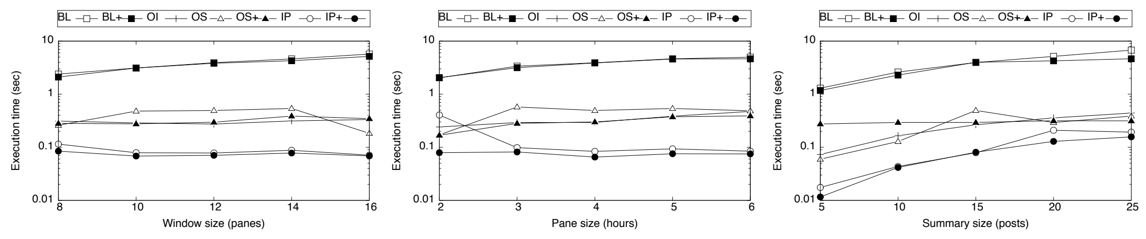
For our experimental evaluation, we have used two real-world datasets from Flickr and Twitter. The first comprises 20 million geotagged images extracted from the publicly available dataset of Flickr photos released by Yahoo! for research [153]. The contained images have worldwide coverage and span a time period of 4 years, from 01/01/2010 to 31/12/2013. Each image is associated with about 6 keywords on average. The second dataset comprises 20 million geotagged tweets. It is the one used in [34], and is also available online[1]. Similar to the Flickr dataset, it also has worldwide coverage. It spans a period of 9 months, from 01/04/2012 to 28/12/2012. The average number of keywords per post is 5.7.

5.6.2 Performance Measures and Parameters

In our experiments, we compare all methods presented in Section 5.4.2, namely Baseline (BL), Online Interchange (OI), Oblivious Summarization (OS), and Intra-Pane Summarization (IP). In addition, for BL, OS, and IP, we also consider their optimized versions as described in Section 5.5. These are denoted in the results by a plus sign (e.g., BL+).

To compare the performance of these methods, we examine two criteria. Firstly, we investigate their efficiency, which is measured as the average execution time required to update the summary every time the window slides. Secondly, we investigate the quality of the summaries they produce, by measuring their objective score (see Definition 5.5). More specifically, we compute this objective score for each summary a given method produces at every slide of the window and we take their average over the entire stream. As discussed earlier, computing the optimal summary (i.e., the one that maximizes the objective function) is practically unfeasible. Thus, to compare the methods to each other, we use the objective score achieved by BL as a reference value and we measure the scores of the rest of the methods as a ratio to that.

[1]http://www.ntu.edu.sg/home/gaocong/datacode.htm



(a) Time vs. window size (m) (b) Time vs. pane size (β) (c) Time vs. summary size (k)

Fig. 5.1 Execution time – Flickr.

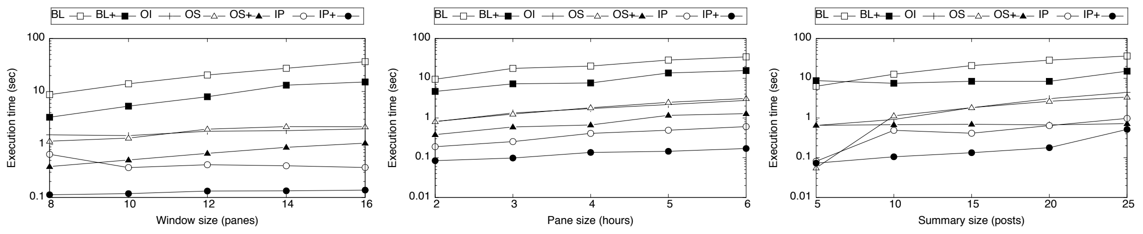
All the algorithms have been implemented in Java and the experiments were conducted on a server with 64 GB memory and an Intel® Xeon® CPU E5-2640 v4 @ 2.40GHz processor, running Debian GNU/Linux 9.0.

To compare the performance of our methods, we process both aforementioned datasets in a streaming fashion, using the sliding window model, as explained in Section 5.3. In our experiments, we set the default pane size to $\beta = 4$ hours. We have chosen a rather large value so that the number of posts contained in the resulting panes is in the order of a few thousands, thus essentially compensating for the fact that these datasets are small samples of the actual stream of posts in these sources. Specifically, the average number of objects per pane is about 2,000 for Flickr and 12,000 for Twitter. Moreover, we set the default window size to $m = 12$ panes and the default summary size to $k = 15$ objects.

In addition, we set both weight parameters α (Equation 5.2) and λ (Definition 5.5) to 0.5, thus weighting equally the spatial and textual dimensions, as well as the two criteria of coverage and diversity. For the IP method, we set the size of each intra-pane summary to $k' = 15$ objects. Finally, for the spatial partitioning used both in computing the spatial coverage (Equation 5.5), as well as in spatio-textual partitioning (Section 5.5.1), we use a uniform grid with resolution 64×64 cells.

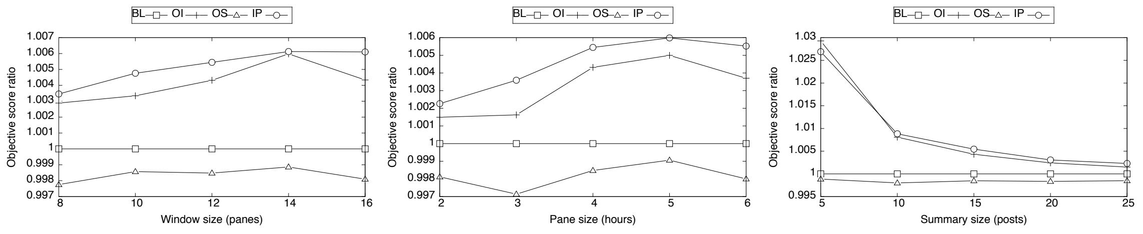
5.6.3 Execution Time

We first examine the execution time of the investigated methods. During these experiments, we vary: (a) the size of the window, in terms of the number m of panes it contains; (b) the size of each pane, in terms of its duration β; and (c) the size of each summary, in terms of the number k of objects it comprises. The respective results are shown in Figure 5.1 for Flickr and in Figure 5.2 for Twitter. Notice that, in these plots, logarithmic scale is used on the y axis.



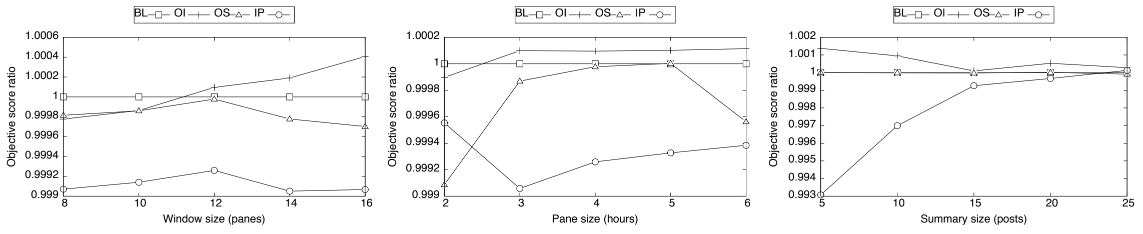
(a) Time vs. window size (m) (b) Time vs. pane size (β) (c) Time vs. summary size (k)

Fig. 5.2 Execution time – Twitter.



(a) Quality vs. window size (m) (b) Quality vs. pane size (β) (c) Quality vs. summary size (k)

Fig. 5.3 Summary quality – Flickr.



(a) Quality vs. window size (m) (b) Quality vs. pane size (β) (c) Quality vs. summary size (k)

Fig. 5.4 Summary quality – Twitter.

With respect to the different strategies, BL appears to have the worst performance in most cases. This is expected, since in this strategy, the previous summary is discarded and the new one is computed from scratch, taking into account all posts in the window. The OI and OS methods outperform BL, having a similar performance to each other. This is because OI constructs the new summary incrementally, discarding only the expired posts from the previous one, and considering only the newly arrived posts as candidates. Similarly, in OS, the benefit results from the fact that although the summary is built from scratch, this is done by only considering the contents of the new pane and the non-expired posts in the previous summary. Yet, IP achieves an even better performance, outperforming all other methods in all experiments. In the case of IP, although all panes are considered, the candidates from the new summary are only drawn from the individual summaries of each pane, i.e., from a significantly smaller pool of posts. Due to this fact, the execution time of IP reduces significantly.

Another important observation concerns the comparison of the performance of the aforementioned methods with their respective optimized versions, employing the spatio-textual partitioning and pruning presented in Section 5.5. The differences here become more apparent in the case of the Twitter dataset, where the amount of posts per pane is about 5 times larger. In this case, the partitioning and pruning technique offers a clear benefit to all methods in which it is applicable, achieving a speedup of about 2 to 5 times.

5.6.4 Objective Score

Next, we investigate the objective score achieved by the different summaries computed by each method. In this set of experiments, we only consider the four different strategies without distinguishing between the optimized and non-optimized versions of each one, since the optimization applied in a strategy only affects its execution time and not the contents of the summary it produces. Moreover, as explained earlier, in each experiment we use the objective score of BL as reference, and we measure the objective scores of the rest of the methods as ratios to that. The results are shown in Figure 5.3 for Flickr and Figure 5.4 for Twitter. As previously, we examine how the results vary for different values of the window size m, pane duration β, and summary size k.

In the Flickr dataset, IP achieves the highest score, followed by OI, and both of them surpass the score of BL. However, all observed differences are rather marginal, not exceeding 1%. In fact, in Twitter, the situation changes, with IP having the lowest score in this case, whereas OI still being slightly better than BL. This noted difference for IP is attributed to the fact that the panes in the case of the Twitter dataset contain a much larger number of objects, thus relying on the intra-pane summaries to select the candidates for the new summary incurs some loss. Nevertheless, again the differences are marginal, implying that in terms of the objective score neither of these strategies appears to have a clear and significant benefit over the others. Subsequently, this leads to the conclusion that one can use the methods offering the lowest execution time without sacrificing the quality of the maintained summary over the stream.

5.7 Summary

In this chapter, we have addressed the problem of continuously maintaining a spatially and textually diversified summary over a stream of spatio-textual documents. We adopt the sliding window model by examining successive chunks of the incoming stream and continuously updating the resulting summary to maximize both the coverage and diversity of its contents. We have formally defined the problem, formulating the criteria for spatio-textual coverage and diversity over the stream of posts, and investigated different strategies that aim at minimizing the computational cost while not sacrificing quality. Moreover, we have proposed specific optimizations that can be applied to further enhance the efficiency of those methods based on spatio-textual partitioning and pruning of posts. Finally, we have experimentally compared the performance of our proposed methods using two real-world datasets from Flickr and Twitter, showing that the proposed optimizations, especially the Intra-Pane Summarization method, can achieve important performance benefits without decreasing the quality of the summary.

CHAPTER 6

DISCOVERY & EXPLORATION OF LOCALLY TRENDING TOPICS

Until now, we have studied problems dealing with the ad hoc and continuous retrieval of objects in the spatial, temporal, and textual dimensions. In this and the subsequent chapter, we exploit the crowdsourced nature of posts to extract patterns, such as hotspots and associations. We start here by discussing the task of finding and exploring local hotspots in the form of trending topics.

6.1 Overview

The sheer volume of content posted on social networks, and its inherent redundancy and noise, makes identifying relevant information or browsing and obtaining an overview of what is happening, challenging and overwhelming. One solution is to restrict the amount of incoming posts and focus on more relevant information. This has been achieved in existing works through research in publish/subscribe systems [31, 30, 161]. Here, user subscriptions in the form of textual, spatial, and/or temporal filters are used to continuously filter out posts according to specified criteria, and either all or a small ranked subset of relevant posts are returned. However, given that social media content often involves new and emerging topics and events, the user may not know in advance what is interesting or relevant, and thus may not be able to specify a suitable geographic area, time period, or keywords for search.

To make it easier for users to get a quick grasp of the most important or interesting information, a common practice is to detect and present to the users a set of popular or trending topics (e.g., sets of hashtags in Twitter) that have high frequency (overall, or currently with respect to the past). However, the popularity of a topic is often not uniformly distributed across space and time; instead, a given topic may only be popular within specific geographic regions and over certain periods of time. In fact, recently there has been a lot of interest in finding local topics and events in Twitter (e.g., [1, 17, 63]). Nevertheless, even if a topic is detected as popular or trending, the posts belonging to it may still be in the order of hundreds or thousands. Hence, besides topic detection, generating topic summaries is also of high importance to let users gain a quick insight into their topics of interest. Similar to topic detection, topic summarization has also received considerable attention in recent years [144, 29, 168].

In this chapter, we present μTOP, a system for discovering and exploring *locally trending topics* in streams of microblog posts. Each topic is represented by a set of one or more keywords (e.g., hashtags in the case of Twitter), and is associated with a *spatio-temporal footprint*, i.e., a set of geographic regions and time periods over which this topic is identified to be popular. Thus, the spatio-temporal evolution of each detected topic is explicitly captured, and can be further explored. In fact, for each of these spatial regions and time intervals for which a topic is popular, μTOP can generate a summary of relevant tweets to describe the topic in more detail.

The remainder of this chapter is structured as follows. The next section outlines our approach and the system architecture, following which Section 6.3 describes in more detail the sub-systems of μTOP. Some usage examples of our application are explained in Section 6.5. Finally, Section 6.6 concludes the chapter.

6.2 Approach and System Architecture

The discovery of locally trending topics is based on the approach presented in [135]. This method segments the space into a uniform grid and detects a set of trending topics in each cell by processing the incoming stream of posts using a sliding window model. Thus, the topics are generated and monitored across space and time as new posts arrive and old ones expire, resulting in an evolving spatio-temporal footprint for each identified topic. To estimate the popularity of a topic, the original approach in [135] counts the number of tweets associated with the topic. However, as we explain later in Section 6.2, our approach, on the other hand, employs the number of distinct users, as opposed to posts, as a measure of a topic's popularity. This has the advantage that it filters out false positive topics made popular via repetitive posts by a single active user or bot.

Moreover, given a topic and its footprint, the system can generate a summary of relevant tweets. For this purpose, the relevant tweets are first retrieved using a spatial-temporal-textual filter, and then the top-k ones are selected according to

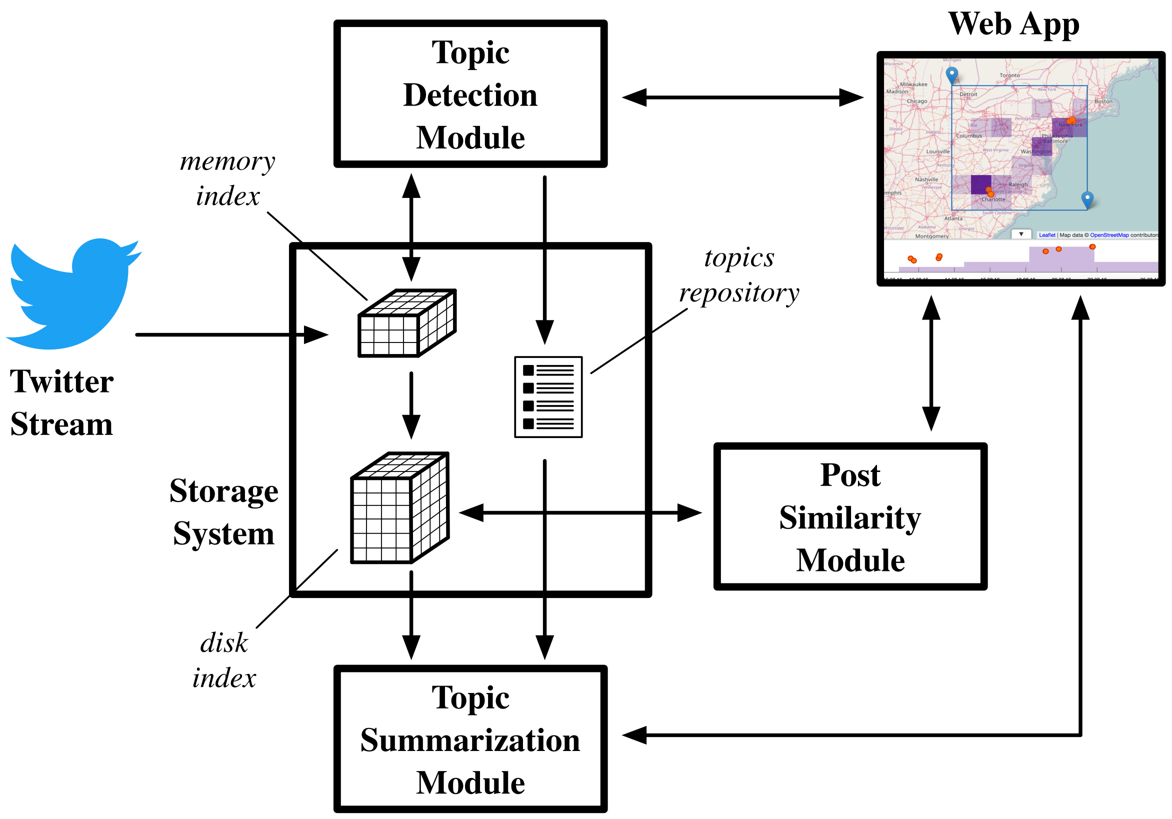


Fig. 6.1 Architecture of μTOP.

the criteria of *coverage* and *diversity*, following the approach presented in [122]. To allow for further exploration, each post can be used to discover other posts based on similarity, by extending the approach presented in [176] to incorporate spatial, textual, and temporal proximity.

Figure 6.1 presents an overview of the architecture of μTOP, which comprises the following main components. The *storage system*, detailed in Section 6.3, is responsible for ingesting the microblog posts (e.g., from Twitter's streaming API), doing some preprocessing (e.g., stemming and stop word filtering), and storing them in main memory and later on disk. This part also maintains all topics and their spatio-temporal footprints. In addition to this, μTOP comprises three core data processing modules: *Topic Detection*, *Topic Summarization*, and *Post Similarity*, which are also discussed in Section 6.3. Moreover, the *Web App*, presented in Section 6.4, consists of the web-based user interface that allows users to issue queries via invoking the appropriate modules and to visualize their results.

6.3 System Modules

We now discuss each of the components of μTOP in more detail, starting with some preliminary definitions that are necessary for our discussion.

6.3.1 Preliminary Definitions

A *post* is represented as a spatial-temporal-textual object $D = \langle u, loc, t, \Psi \rangle$, where u is the identifier of the user making the post, $loc = (x, y)$ is the post's geolocation, t is the post's timestamp, and Ψ is a set of keywords representing the post's textual content.

We also need to define textual, spatial, and temporal distance functions between posts. Given two posts D_i and D_j, their *textual distance* δ_ψ is measured by the Jaccard similarity between their keyword sets:

$$\delta_\psi(D_i, D_j) = 1 - \frac{|D_i.\Psi \cap D_j.\Psi|}{|D_i.\Psi \cup D_j.\Psi|}.$$

The spatial and temporal distances are measured, respectively, by the Euclidean distance d between the posts' locations and the time difference between the posts' timestamps. To be able to aggregate distance scores across dimensions, we normalize spatial and temporal distances to values in the range $[0,1]$ (notice that $\delta_\psi \in [0,1]$). For that purpose, we assume that the posts under consideration are enclosed by a bounding box with diameter length γ and a time interval of length τ. Then, we define the (normalized) *spatial distance* δ_s and *temporal distance* δ_t as follows:

$$\delta_s(D_i, D_j) = \frac{d(D_i.loc\,, D_j.loc)}{\gamma}\ ,\ \delta_t(D_i, D_j) = \frac{|D_i.t - D_j.t|}{\tau}.$$

6.3.2 Storage System

We now explain the indexing strategy adopted by μTOP. To allow for efficient real-time detection of locally trending topics and the exploration (retrieval, summarization) of past topics and posts, we adopt a hybrid data indexing structure, involving both the main memory and the disk. This structure, depicted in Figure 6.2, indexes along all four attributes, latitude, longitude, time, and text. A 3-dimensional grid provides access along the first three attributes, while within each cell an inverted index provides efficient retrieval by keyword.

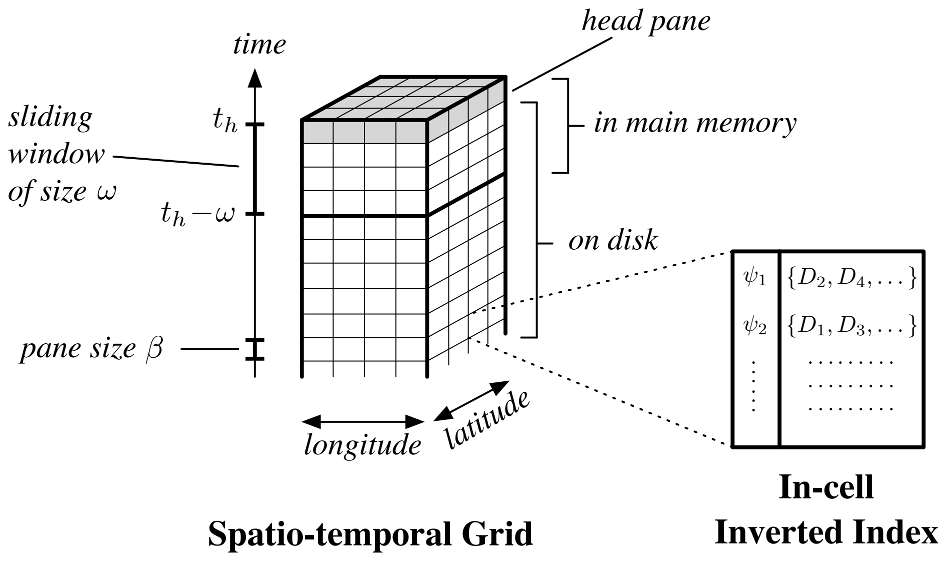


Fig. 6.2 Overview of indexing scheme in μTOP.

Each grid cell has size $g \times g \times \beta$, where g is a fixed arc range (for latitude and longitude) partitioning the world (or the spatial area of interest) and β is a fixed time interval. The inverted index of each cell associates each keyword with a list of posts in that cell that contain it. A slice of the grid in the temporal dimension containing posts that were published in an interval of β time units (e.g., one hour) is called a *pane*. The pane collecting the most recent posts is called the *head pane*.

The main memory index only stores the latest ω/β panes, and thus indexes posts that were published within a *sliding window* of ω time units (e.g., one day) in the past. This part of the grid is used by the topic detection module (Section 6.3.3). On the other hand, the disk-based index stores all panes except the head. This index is used by the topic summarization and the post similarity modules (Sections 6.3.4 and 6.3.5).

Besides this hybrid index structure, the storage system of μTOP includes a repository archiving all trending topics, along with their spatio-temporal footprints. The repository receives the continuous output of the topic detection module and provides input to the topic summarization module when requested.

6.3.3 Topic Detection

In μTOP, *topic detection* is based on the work presented in [135]. We briefly describe the main aspects of the process below.

To process the incoming stream of posts, a lightweight, in-memory spatial index comprising a uniform spatial grid is used, as explained in Section 6.3.2. Upon arrival, each incoming post D is assigned to the corresponding grid cell c according to its geolocation $D.loc$. In each cell, the local stream of posts is processed to generate and maintain locally popular topics with respect to a sliding window W of range ω and sliding step β.

A topic C is characterized by a set of keywords (e.g., hashtags) $C.\Psi$ and is associated with the grid cell c and the time window W in which it is detected. The *popularity* $C.pop$ of a topic C within the cell c and time window W is determined by the number of users having posts in c and W that textually match this topic. We say that a post D matches a topic C if their textual similarity $\delta_{\psi}(D.\Psi, C.\Psi)$ is above a specified threshold $\theta_{\psi} \in [0,1]$. The popularity score of a topic is normalized by the total number of users having posts within the cell c and window W. If an incoming post does not match any of the existing topics in the current cell and time window, a new topic is created having as keywords those appearing in this post. Eventually, those topics with popularity higher than a specified threshold $\theta_u \in [0,1]$ are marked as *locally trending*, and are returned.

If the same topic is detected in multiple cells and/or time windows, these are merged to construct the topic's *spatio-temporal footprint* $C.\mathcal{F} = \{(c_i, W_i)\}$. Hence, this process not only detects locally popular topics, but also explicitly associates each one with the exact geographic region(s) and time period(s) within which it is popular.

6.3.4 Topic Summarization

Once topics are detected, the next step is to get a summarized overview of each topic. A summary of a topic is already provided by the set of keywords defining it and its spatio-temporal footprint. However, a list of representative posts may also be needed in order to describe the topic in more detail.

For this purpose, μTOP can generate a summary, comprising k posts, for any part of the topic's spatio-temporal footprint. In other words, it can compute a set of k representative posts for any region and time window in which the given topic has been popular. The size of each summary, i.e., the value of the parameter k, can be specified by the user, and can be different for each summary.

The selection of the k representative posts to be included in the summary is based on the criteria of *coverage* and *diversity*. In particular, each summary is constructed by executing a *Coverage & Diversity Aware Top-k Spatial-Temporal-Keyword* (kCD-STK) query, following the approach presented in [122]. We outline the main aspects of this process next.

Formally, a kCD-STK query is defined by a tuple of the form $Q = \langle R, T, \Psi, k \rangle$, where R is a spatial region, T is a time interval, Ψ is a set of keywords, and k is the number of results to return. In our case, the filters R, T, and Ψ are derived from the topic's keyword set and spatio-temporal footprint, while k is determined by the desired summary size. The distinguishing aspect of the kCD-STK query is that instead of selecting the top-k posts ranked by relevance, it selects a more representative set of k posts using the criteria of coverage and diversity, which are defined below.

Let $\mathcal{D}_F$ denote the set of all posts satisfying the spatial, temporal, and textual filters R, T, and Ψ in the query Q. The *coverage* of a post $D \in \mathcal{D}_F$ is defined as the ratio of relevant posts that are within spatial distance θ_s and temporal distance θ_t from D, i.e.,

$$cov(D, \mathcal{D}_F) = \frac{|\{D' \in \mathcal{D}_F : d_s(D, D') \leq \theta_s \wedge d_t(D, D') \leq \theta_t\}|}{|\mathcal{D}_F|}.$$

This is a measure of how representative this particular post is with respect to other relevant posts. Moreover, this is extended to measure the coverage of a set of selected posts $\mathcal{R} \subseteq \mathcal{D}_F$ of size k:

$$cov(\mathcal{R}, \mathcal{D}_F) = \frac{1}{k} \sum_{D \in \mathcal{R}} cov(D, \mathcal{D}_F).$$

Essentially, the criterion of coverage favors the selection of posts from locations that contain a large number of relevant posts.

On the other hand, to avoid a high degree of redundancy, the criterion of *diversity* is used to increase the dissimilarity among the selected posts. Specifically, the diversity of a pair of posts $D_i, D_j \in \mathcal{D}_F$ is defined as:

$$div(D_i, D_j) = \alpha \cdot d_s(D_i, D_j) + (1 - \alpha) \cdot d_t(D_i, D_j),$$

where $\alpha \in [0, 1]$ is an adjustable weight parameter between the spatial and the temporal distances. Furthermore, the diversity of a set of posts $\mathcal{R} \subseteq \mathcal{D}_F$ of size k is calculated as:

$$div(\mathcal{R}) = \frac{1}{k \cdot (k-1)} \sum_{D_i, D_j \in \mathcal{R}, i \neq j} div(D_i, D_j).$$

Based on the above, the kCD-STK query returns a set of k posts $\mathcal{R}^*$ that maximizes a combined measure of coverage and diversity:

$$\mathcal{R}^* = \underset{\mathcal{R} \subseteq \mathcal{D}_F, |\mathcal{R}|=k}{\arg\max} \ \{(1-\lambda) \cdot cov(\mathcal{R}, \mathcal{D}_F) + \lambda \cdot div(\mathcal{R})\},$$

where $\lambda \in [0,1]$ is a parameter determining the trade-off between maximum coverage ($\lambda = 0$) and maximum diversity ($\lambda = 1$).

6.3.5 Retrieving Similar Posts

The above process provides a flexible and adjustable way to get a summary of representative and diverse posts for a topic across the whole extent of its spatio-temporal footprint. Then, the user can further drill down into the topic by selecting any of the posts in the presented summary that seems interesting and requesting other similar posts to it. That is, the posts contained in each summary can serve as *seeds* for further exploration of the topic's contents.

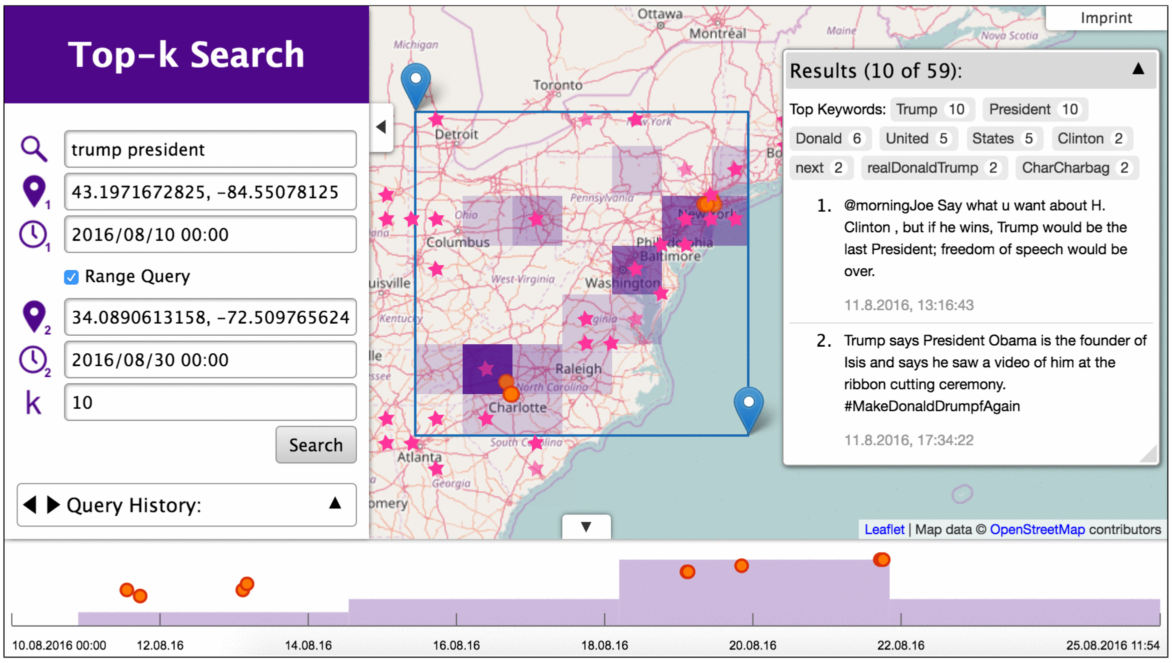
This is performed by executing a top-k spatial-temporal-keyword query $Q = \langle loc, t, \Psi, k \rangle$, where loc, t, and Ψ are, respectively, the location, the timestamp, and the keyword set of the selected post D, and k is the number of similar posts to be retrieved. Here, Q can be regarded as an extension of the standard top-k spatial keyword query to incorporate temporal information. Thus, in this case, the query returns the top-k results ranked by *relevance* determined by an aggregate distance score δ combining the partial distance scores in the spatial, temporal, and textual dimensions. The distance score used in μ`TOP` is shown below:

$$\delta(D, D') = w_s \cdot \delta_s(D, D') + w_t \cdot \delta_t(D, D') + w_\psi \cdot \delta_\psi(D, D')$$

where $w_s \in [0,1]$, $w_t \in [0,1]$, and $w_\psi = 1 - w_s - w_t$ are weights determining the relative importance of each distance score.

6.4 User Interface

The user interface of our prototype is shown in Figure 6.3. The map continuously depicts locally trending topics as discovered by the topic detection module. Topics are shown as stars, with brightness indicating popularity. Hovering over a star reveals the topic's spatial footprint, whereas clicking on it shows its keywords together with two options (Figure 6.6 left). The first option is to invoke the *post similarity* module



Top-k Search

trump president

43.1971672825, -84.55078125

2016/08/10 00:00

Range Query ✓

34.0890613158, -72.509765624

2016/08/30 00:00

k 10

Search

Query History:

Results (10 of 59):

Top Keywords: Trump 10 | President 10 | Donald 6 | United 5 | States 5 | Clinton 2 | next 2 | realDonaldTrump 2 | CharCharbag 2

1. @morningJoe Say what u want about H. Clinton , but if he wins, Trump would be the last President; freedom of speech would be over.

   11.8.2016, 13:16:43

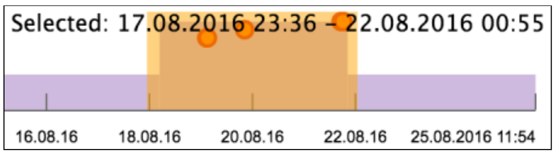
2. Trump says President Obama is the founder of Isis and says he saw a video of him at the ribbon cutting ceremony. #MakeDonaldDrumpfAgain

   11.8.2016, 17:34:22

Fig. 6.3 The user interface showing the results of a summarization request.

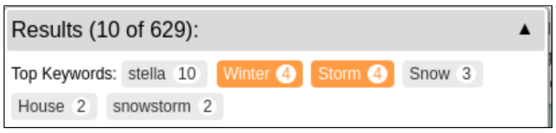
to retrieve a ranked list of similar posts (in terms of spatial proximity, time closeness, and textual relevance). The resulting posts are displayed in a pop-up window on the right, and also as orange dots on the map and on the timeline located at the bottom.

The second option for a locally trending topic is to explore its spatio-temporal footprint by invoking the *topic summarization* module. The sidebar on the left displays a form detailing the spatial and temporal ranges for the summary, as well as the keywords and the number of returned results. The default number of results returned is ten. Naturally, the user can specify her own summarization request by changing the values in the form. The summarization results are listed in a pop-up window on the right, where the user can filter them by the top keywords shown at the top (Figure 6.4(b)). The spatial and temporal distributions of the results are shown on the map and on a timeline at the bottom using orange bullets, respectively, as depicted



Selected: 17.08.2016 23:36 – 22.08.2016 00:55

(a) Timeline (selected range in *orange*).



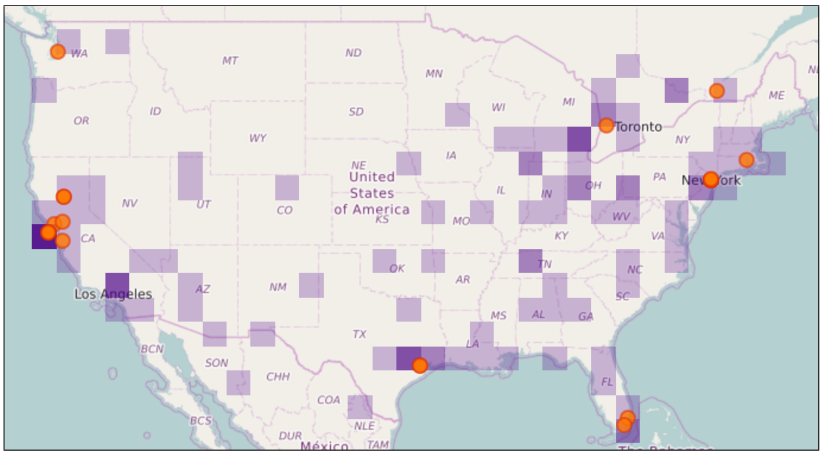
Results (10 of 629):

Top Keywords: stella 10 | Winter 4 | Storm 4 | Snow 3 | House 2 | snowstorm 2

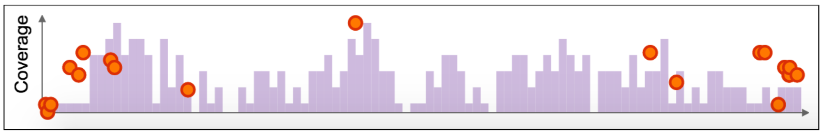
(b) Top keywords (selected keywords in *orange*).

Fig. 6.4 Filtering summarization results by top keywords and temporal range.

in Figure 6.5. The height of the purple bars in the timeline indicates the average coverage in the corresponding temporal range. Similarly, the purple rectangles on the map illustrate the average coverage in the corresponding regions. The darker the color, the higher the coverage in the area.



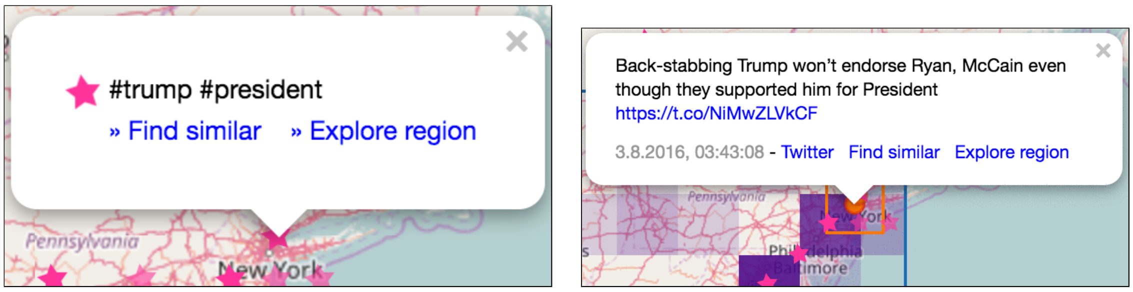
(a) Spatial distribution.



(b) Temporal distribution.

Fig. 6.5 Spatial and temporal distributions of summarization results.

Further exploration of the topic summarization results is provided by two means. First, the timeline allows the user to filter the results by selecting a temporal sub-range (Figure 6.4(a)). This issues a new topic summarization request using the sub-range and updates the results. Second, by clicking on a result on the map, besides showing its content and a link to the post, μTOP displays two additional links (Figure 6.6 right). The one issues a *retrieve similar posts* request using the result's attributes, while the other allows the user to further explore the highlighted spatio-temporal region issuing a new *topic summarization* request. Again, the results are shown on the map and on a timeline. In the case of a post similarity request, the timeline additionally shows the query timestamp as a gray vertical line. Finally, we can browse through the executed queries using the history at the bottom of the sidebar (Figure 6.3).



Back-stabbing Trump won't endorse Ryan, McCain even though they supported him for President https://t.co/NiMwZLVkCF

3.8.2016, 03:43:08 - Twitter

Fig. 6.6 A locally trending topic and a post summarizing it.

6.5 Demonstrating Example

To demonstrate the efficiency and effectiveness of μTOP, tweets are continuously being collected from the public Twitter Streaming API[1]; the current dataset contains over 80 million geotagged tweets with worldwide coverage. The topics are monitored on a stream arriving at an average rate of approximately 500,000 tweets per day. A live demo[2] of μTOP is available online, accompanied by a video[3] explaining and demonstrating its functionality.

Next, we outline a typical usage scenario for demonstration. Initially, the user interface shows locally trending topics on a map, depicted by star icons. Clicking on a star icon reveals the topic's hashtags, for example "#trump #president", as shown in Figure 6.6. The *Explore region* link is then used to summarize the topic. It issues a topic summarization request that displays the resulting tweets in a list, on the map, and on the timeline. Alternatively, the user may enter query parameters manually using the form in the sidebar on the left, for example, to increase the spatial area and time interval. The same form can also be used to directly issue a post similarity request by unchecking the *Range Query* option and specifying only a single location and point in time.

At the top of the result list a set of keywords is shown that are popular in the result set. This reveals new keywords that are frequently used together with the query keywords *Trump* and *President*. For example, *Clinton* is used in 20% of the results. We can click on it to view only those posts that contain this word.

When a topic is summarized, the average coverage is shown as purple blocks and bars in addition to the results. This allows to easily identify spatial regions and time intervals where the topic is popular. For example, Figure 6.3 shows that the

[1]https://dev.twitter.com/streaming/public
[2]http://mtop.imp.fu-berlin.de
[3]https://youtu.be/OmXJUGndaQA

topic is popular around New York City and between the 18th and 22nd of August. This spatial region and time interval can be further explored by issuing another topic summarization request, for example, by moving the blue markers on the map or by selecting a temporal range on the timeline. We can return to the previous result set by clicking the back-arrow button in the *Query History*, shown in the sidebar.

Instead of summarizing a particular topic, we can also explore a topic by invoking a post similarity search without limiting the spatial and temporal range. By clicking the *Find similar* link, a list of posts similar in spatial, temporal, and textual content is compiled.

6.6 Summary

In this chapter, we have presented μTOP, a system for detecting and exploring locally trending topics in microblog posts based on spatial, temporal, and textual criteria. Using a sliding window over an incoming stream of posts, μTOP detects locally trending topics, and associates each one with a spatio-temporal footprint. Then, for each spatial region and time period in which a certain topic is trending, the system generates a summary of the relevant posts, by selecting top-k posts based on the criteria of coverage and diversity. μTOP includes a Web-based user interface, providing a comprehensive way to visualize and explore the detected topics and their spatio-temporal summaries via a map and a timeline. The functionality of the system has been demonstrated using a continuously updated dataset containing more than 80 million geotagged tweets and by going through a typical usage scenario.

Chapter 7

Mining Associated Location Sets

In the previous chapter, we presented a system for the detection and exploration of trending topics in social networks. Here, we utilize posts for a different type of analysis, namely the discovery of associations between places.

7.1 Overview

In this chapter, we seek to find *Socio-Textual Associations* (STAs) among locations that are strongly supported by a corpus of geotagged posts. Given a set of keywords, we say that a group of locations are socio-textually associated if a user has posts near each of these locations and the combined keyword set of these posts contains all query keywords. The more people make an association, i.e., the stronger its support in the corpus is, the likelier it is that there exists a latent thematic connection among the locations.

Compared to previous works that search for connections among a group of locations, our work has the distinguishing and novel aspect that it considers *social and textual criteria in unison* to define associations. The social condition ensures that the locations co-occur in user trails, while the textual requirement ensures that users have made posts that are collectively relevant to the query keywords at these locations. In location-based services, given a complex information need (typically expressed by a query comprising multiple keywords) it is often possible that no single object or location satisfies all query keywords. To address this, *Collective Spatial Keyword* (CSK) queries have been proposed and studied in the literature (refer to Section 2.2.2 for an overview). These queries return sets of locations that collectively cover all query keywords and are spatially close to each other. Thus, the locations are grouped according to textual criteria (keyword coverage) as well as spatial criteria (spatial proximity). In other words, for a given a set of keywords, the optimization objective is an aggregate spatial distance, instead of some evidence-based frequency metric, and the strength of the association among a valid group of locations (i.e., one that covers all keywords) is defined by spatial proximity alone. The intuition behind this grouping is that users are more likely to visit locations that are close to each other. Although this assumption is true in many cases, especially when users have a limited time budget, it fails to establish a thematic connection evidenced by users' behavior. For example, the fact that there is a restaurant next to an art exhibition venue, does not necessarily imply that art-loving people would find this particular restaurant attractive, unless such a connection is indeed supported by a large number of posts, from the same users, containing, for example, both keywords "*art*" and "*restaurant*" around these locations. As a matter of fact, if a strong thematic association among nearby locations exists, our problem formulation will certainly capture it.

In another line of work (e.g., [102, 98, 147, 18, 169, 181]), which we term *Location Patterns* (LP), the objective is to determine groups, patterns, or sequences of locations (or regions) that are frequent in terms of purely social criteria, i.e., how many people support them. Since the process ignores the textual aspect, the identified locations are not semantically characterized or distinguished, and thus there is no mechanism to explore or exploit the resulting groups under a thematic context. For instance, this limits queries to finding the overall most frequent sequence of locations in a given area or the most frequent POI to visit next. Even though one could easily enrich locations with textual information *after* the mining process, say to support recommending the most frequent restaurant to visit next, the locations remain only socially associated, and not thematically, because the computed frequencies still ignore the textual aspect.

A rather straightforward way to associate locations with keywords according to users' behavior is based on rank aggregation [53]. For each keyword, consider a ranking of locations according to the keyword popularity, i.e., the number of posts that contain it. Then, to derive a group of locations that is most associated with a set of keywords, one can simply collect the most popular location for each keyword. This approach, which we call *Aggregate Popularity* (AP), has the advantage that individual locations are strongly associated with their respective keywords, but the location set as a whole may lack a strong socio-textual association. Indeed, each location may be popular for a different type of users, hence there may be no significantly sized population for which all these locations are popular. Exactly as in the case

Table 7.1 Categorization of existing work and ours (STA).

Line of Work	Information Exploited			Optimization
	Spatial	Textual	Social	Objective
Location Patterns (LP) [18, 98, 102, 147, 169, 181]	×		×	frequency
Collective Spatial Keyword (CSK) [21, 174]	×	×		proximity
Aggregate Popularity (AP)	×	×	×	popularity
Socio-Textual Associations (STA)	×	×	×	frequency

of proximity-based associations, if a strong thematic association among popular locations exists, our socio-textual approach will discover it.

Another differentiating trait of our work is that we consider the textual information that is included in the posts themselves and do not rely on an external categorization of locations or POIs. The reason is that we seek to exploit the *wisdom of the crowd* to also determine textual relevance, in addition to quantifying the strength of derived associations. Nonetheless, our methods can be readily adapted to take into account external textual descriptions as well.

To better frame our contribution with respect to previous works, Table 7.1 summarizes all approaches according to the type of information they exploit, i.e., spatial, textual, or social (user id), as well as the objective they optimize for. Mining location patterns does not exploit textual information, and seeks for groups of locations that maximize the frequency with which they co-appear among users' trails. On the other hand, collective spatial keyword queries ignore the social aspect, and look for location sets that maximize their proximity (to each other and/or to a target location) subject to the constraint that they cover given keywords. An approach based on aggregating popularity considers all types of information available, and strives to include locations that are individually popular for some keyword and collectively cover given keywords. Our work also considers all types of information, but optimizes for a frequency metric that counts co-appearances of locations under a certain theme/topic/context, which is defined by the given keywords.

As an example, consider a search for locations in Berlin using the keywords "*wall*", "*art*", and "*restaurant*". Figure 7.1 depicts the results returned by different alternative approaches for combining locations to satisfy these keywords. Our socio-textual based approach returns the following location set as the top result (star-shaped markers): ⟨ "East Side Gallery", "Hackescher Markt" ⟩. The former is a portion of the Berlin wall covered with paintings, hence hosting many posts with the keywords "*wall*" and "*art*". The latter is a popular square in the city center, hosting also a series of restaurants

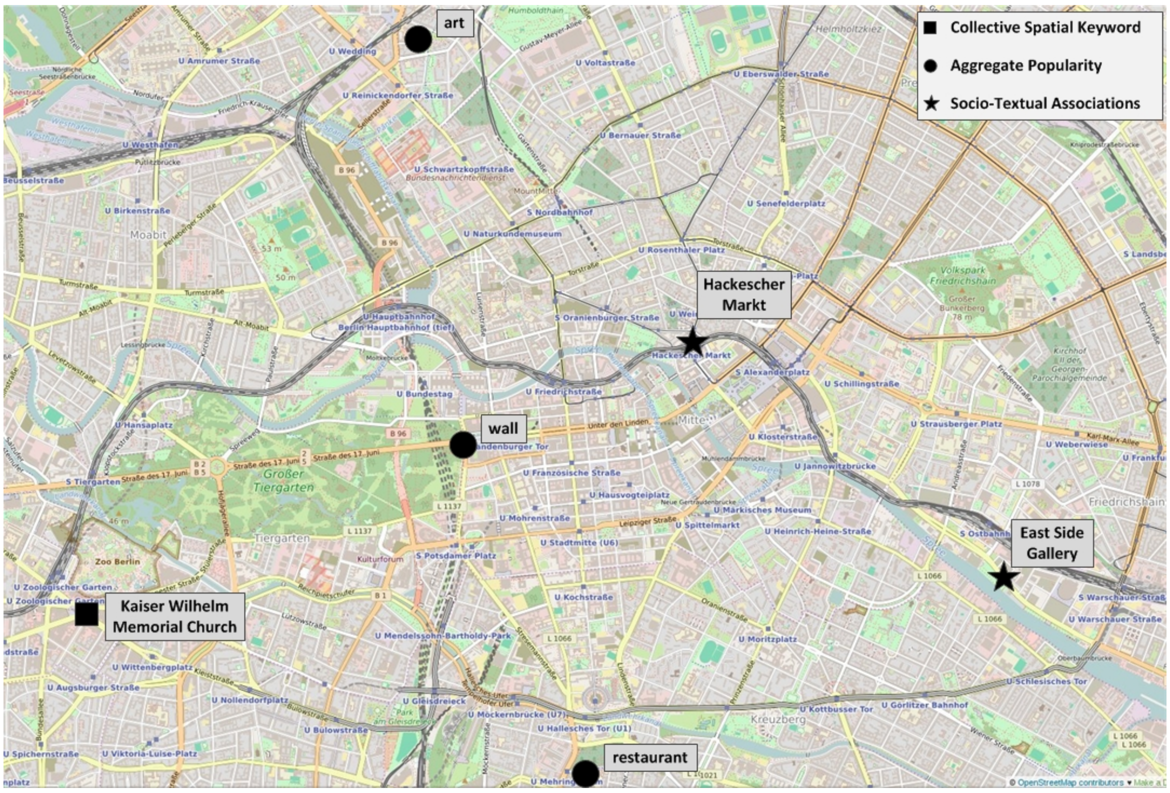

■ Collective Spatial Keyword
● Aggregate Popularity
★ Socio-Textual Associations
art
wall
restaurant
Hackescher Markt
East Side Gallery
Kaiser Wilhelm Memorial Church

Fig. 7.1 Example of location sets retrieved for keywords "*wall*", "*art*", and "*restaurant*" in Berlin.

frequently visited by tourists and travelers. As it turns out, these locations are neither the most popular ones for each individual keyword (see locations with circle-shaped markers, returned by the AP approach) nor close to each other. Yet, they reveal an interesting association, hinting to the fact that many travelers that have visited or plan to visit the Wall, being interested in art, tend to also prefer restaurants located at Hackescher Markt.

Furthermore, a search based on CSK query identified around 350 singleton locations, for which there exists at least one user with posts containing all query keywords. One of these results is illustrated in Figure 7.1 (square-shaped marker). It is not straightforward how to select the best among these results; in fact, several of them may even be due to outliers or noise, which are inherent to crowdsourced content. Since a CSK query does not take frequency into account, it is better suited for cases where the query terms refer to (curated) POI categories, while being error prone and sensitive to outliers when searching on raw tags. On the other hand, the top result based on AP consists of Brandenburg Gate (for "*wall*"), a famous monument close to where the Berlin wall used to pass; the intersection of Gneisenaustr. and

Mehringdamm streets (for "*restaurant*"), a place with many popular restaurants; and Stattbad Wedding (for "*art*"), a former well-known art venue. Each of these locations is popular for the respective query keyword, but they do not represent any strong shared interest between the people visiting them.

Existing algorithms for related problems cannot be used to extract socio-textual associations. Although our problem seems similar to mining frequent location patterns, the requirement for the locations to collectively cover certain keywords significantly complicates the problem, as we discuss in Section 7.4. Specifically, our notion of support (frequency) for a location set *does not exhibit the anti-monotonicity property* necessary to apply an Apriori-like algorithm [2]. Briefly, such a property would allow for early pruning of location sets that cannot be extended to produce valid results. Practically, the implication is that a naïve algorithm for even a relatively small-sized city-level dataset, with around 20,000 distinct locations, would need to investigate more than 10^{12} sets of three locations.

Nevertheless, by studying the problem characteristics, we are able to introduce a weaker notion of support that (1) exhibits anti-monotonicity, and (2) is an upper bound on the actual support of location sets. Armed with these two properties, we then introduce a methodology to efficiently identify location sets with strong socio-textual associations. Moreover, we study three different implementations of this methodology, each having its own merits. In the simplest, we assume that no pre-processing is allowed and that no index structure is available. We then present a method based on a simple off-the-shelf inverted index, and demonstrate how it can significantly speed up processing. The only caveat is that the association of locations with nearby posts is assumed to be known beforehand. Finally, leveraging the recent advances in spatio-textual indexes, we devise an algorithm that exploits their general functionality. In particular, we consider the state-of-the-art I^3 index [175], which we also extend further to derive an even faster approach. Compared to the inverted index approach, the spatio-textual index methods allow to define the association of locations with nearby posts dynamically, which causes an overhead in execution time but provides higher flexibility.

In addition, we consider the problem of ranking socio-textually associated location sets instead of relying on a user-specified minimum support threshold. Thus, we directly address the problem of identifying the k most strongly associated location sets. We describe a general methodology, and then propose algorithms that build upon their threshold-based counterparts.

The main contributions of this chapter are summarized below:

- We introduce and formally define the problem of finding socio-textually associated location sets.
- We study the problem characteristics and introduce a general framework based on a weaker support measure, which satisfies the desirable anti-monotonicity property.
- We present a basic algorithm, and two efficient algorithms that exploit an inverted index and a spatio-textual index, respectively, to significantly speed up computation.
- We consider the ranking variant of the problem and discuss the necessary adaptations to all proposed algorithms.
- We present results from an experimental evaluation using real-world data from geolocated Flickr photo trails in three major cities.

The remainder of this chapter is organized as follows. In the next section, we present related work on mining mining frequent locations from geotagged posts. Then, we formally define the problems in Section 7.3 and study their characteristics in Section 7.4. Following this analysis, we present our algorithms in Section 7.5 and extend them to the top-*k* variant in Section 7.6. Finally, Section 7.7 presents our experimental evaluation and Section 7.8 concludes the chapter.

7.2 Additional Relevant Background

Having provided an overview of our problem, we now discuss some of the relevant works in the area of mining frequent locations from geotagged posts. There are several approaches that analyze trails of geotagged posts, mainly photos, to extract interesting Location Patterns (LP), such as scenic routes or frequently traveled paths. A typical methodology is to use a clustering algorithm to extract landmark locations from the original posts, and then apply sequence pattern mining.

In [102], clustering is first used to identify POIs; then, association rule mining is applied to extract associative patterns among them. In [98], each photo is first assigned to a nearby POI, whereas, for the remaining ones, a density-based clustering algorithm is applied to generate additional locations. Then, a travel sequence is constructed for each user and sequence patterns are mined from these individual travel sequences. In [147], kernel vector quantization is used to find clusters of photos;

then, routes are defined as sequences of photos from the same user and patterns are revealed by applying hierarchical clustering on routes using the Levenshtein distance. In [18], a trajectory pattern mining algorithm is applied on geotagged Flickr photos to identify frequent travel patterns and regions of interest. In [148], a clustering method is applied on geotagged photos to identify and rank popular travel landmarks.

Geotagged photos have been used to measure the attractiveness of road segments in route recommendation. A tree-based hierarchical graph is used in [180] to infer users' travel experiences and interest of a location from individual sequences. Considering the transition probability between locations, frequent travel sequences are identified. Ranking trajectory patterns mined from sequences of geotagged photos is investigated in [169]. The mean-shift algorithm extracts locations from the original GPS coordinates of the photos; then, the PrefixSpan algorithm identifies the frequent sequential patterns, which are ranked based on user and location importance. In [181], density-based clustering is used to identify regions of attractions from trails of geotagged photos; then, the Markov chain model is applied to mine transition patterns among them.

Other efforts have focused on automatic trip planning or personalized scenic route recommendations based on geotagged photo trails, taking into account user preferences, current or previous locations, and/or time budget (e.g., [113, 149]). In [45], individual photo streams are integrated into a POI graph and itineraries are constructed based on POI popularity, available time, and destination. In [118], users' traveling preferences are learned from their travel histories in one city, and then used to recommend travel destinations and routes in a different city. In [100], a set of location sequences that match the user's preferences, present location, and time budget are computed from individual itineraries. From a different perspective, a Bayesian approach is applied in [11] to test different hypotheses about how photo trails are produced. Various assumptions are assessed, e.g., that users tend to take photos close to the city center, near POIs, close to their previous location, or a mixture of these. Finally, in a different direction, a classification method for predicting the location of photos based on visual, textual, and temporal features is presented in [43]. Then, these photos are used to automatically identify places that people find interesting. Furthermore, the proposed method selects representative photos to describe places.

Similar to the works presented above, we also select locations that appear frequently in users' posts. However, in our case these locations should be strongly associated with a given set of keywords, a requirement which complicates the search.

7.3 Model and Definitions

Assume a database of posts $\mathcal{P}$ made by users $\mathcal{U}$. Each post $p \in \mathcal{P}$ is a tuple $p = \langle u, \ell, \Psi \rangle$, where $p.u \in \mathcal{U}$ is the user that made the post, $p.\ell = (lon, lat)$ is the geotag (location) of the post, and $p.\Psi$ is a set of keywords that characterize it. We use $\mathcal{P}_u$ to denote all posts of user u, i.e., $\mathcal{P}_u = \{p \in \mathcal{P} : p.u = u\}$. Furthermore, assume a database of locations $\mathcal{L}$. These may correspond to the posts' locations or, for generality, may also be defined independently of $\mathcal{P}$. For instance, one may use a POI database to populate $\mathcal{L}$, or apply a clustering algorithm on the posts' geotags and then construct $\mathcal{L}$ from the cluster centroids. Thus, we reserve the term *location* for a member of $\mathcal{L}$ and refer to a post's location as its *geotag*. Table 7.2 summarizes the most important notation.

Locations are the principle objects in our problem. We seek to identify sets of locations that are *strongly associated* with a set of keywords. To define this association, we first introduce the concepts of locality and (textual) relevance for a post.

Definition 7.1 (Local Post). *A post p is* local *to location ℓ if the post's geotag is within distance ϵ to ℓ, i.e., if $d(p.\ell, \ell) \leq \epsilon$, where d is a distance metric (e.g., Euclidean).*

Definition 7.2 (Relevant Post). *A post p is* relevant *to keyword ψ if the post's keyword set contains ψ, i.e., if $\psi \in p.\Psi$.*

Posts associate locations with keywords. These associations are bestowed by users themselves, as opposed, for example, to a specific POI categorization made by a particular source; thus, they capture the wisdom of the crowd. To model the relationships between users' posts, locations, and keywords, we introduce a bipartite graph, where the two types of vertices correspond to keywords and locations, while edges correspond to users' posts.

Definition 7.3 (Association Graph). *The* Association Graph *is a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \Psi \cup \mathcal{L}$ and $\mathcal{E} \subseteq \Psi \times \mathcal{L}$, such that an edge $e = (\psi, \ell)$ exists iff there exists at least one post p which is local to ℓ and relevant to ψ; moreover, e is labeled with the set of users that have made such posts.*

Figure 7.2 shows a running example with the posts of five users $u_1, \ldots, u_5$ around three locations ℓ_1, ℓ_2, ℓ_3, containing two keywords ψ_1, ψ_2. Post p_{ij} denotes the j-th post of the i-th user. For instance, post $p_{12} = \langle u_1, \ell_2, \{\psi_1, \psi_2\} \rangle$ of user u_1 is local to location ℓ_2 and relevant to keywords ψ_1 and ψ_2. The resulting Association Graph is depicted in Figure 7.3.

Table 7.2 Summary of notation for STA.

Symbol	Definition
p, $\mathcal{P}$	post, database of posts
u, $\mathcal{P}_u$	user, posts of user
ℓ, L, $\mathcal{L}$	location, set of locations, database of locations
ψ, Ψ	keyword, set of keywords
$\mathcal{U}_{L\Psi}$	set of users supporting (L, Ψ)
$\mathcal{U}_{L\widetilde{\Psi}}$	set of users weakly supporting (L, Ψ)
$\mathcal{U}_{\Psi}$	set of users relevant to Ψ
$sup(L, \Psi)$	support of (L, Ψ)
$w\_sup(L, \Psi)$	weak support of (L, Ψ)
$rw\_sup(L, \Psi)$	relevant and weak support of (L, Ψ)
σ	support threshold

The association between a keyword and a location is explicit and its strength can be quantified by the number of users making it. For example, three users have associated keyword ψ_1 with location ℓ_3 in the running example. On the other hand, the association between sets of keywords and sets of locations is not immediately apparent, e.g., what the textual description of the location set $\{\ell_1, \ell_2\}$ should be. If it is simply the set of keywords that have an edge towards the location set, then how do we quantify its strength if different users have made different associations? The location set should be strongly associated with a set of keywords not because there exist edges with multiple users in the Association Graph, but because there exists *a large number of users that agree on this association*. Therefore, the key question to answer is when a user supports an association between a location set and a keyword set.

Definition 7.4 (Supporting User). *A user u* supports *the association between a location set L and keyword set Ψ, denoted as $u \in \mathcal{U}_{L\Psi}$, if:*

- *for each keyword $\psi \in \Psi$, the user has made a post relevant to ψ and local to a location $\ell' \in L$, i.e., every $\psi \in \Psi$ is connected via a u-labeled edge to some $\ell' \in L$; and*
- *for each location $\ell \in L$, the user has made a post local to ℓ and relevant to a keyword $\psi' \in \Psi$, i.e., every $\ell \in L$ is connected via a u-labeled edge to some $\psi' \in \Psi$.*

Hence, a user supports association (L, Ψ) if her posts connect each keyword in Ψ to some location in L and, vice versa, each location in L to some keyword in Ψ. This implies a tight coupling between *all* keywords and *all* locations, according to the user.

		Locations	
Users	ℓ_1	ℓ_2	ℓ_3
u_1	$p_{11} : \{\psi_1\}$	$p_{12} : \{\psi_1, \psi_2\}$	$p_{13} : \{\psi_1\}$
u_2	$p_{21} : \{\psi_1\}$	$p_{22} : \{\psi_1\}$	
u_3	$p_{31} : \{\psi_2\}$	$p_{32} : \{\psi_1\}$	$p_{33} : \{\psi_1\}$
u_4		$p_{42} : \{\psi_2\}$	$p_{43} : \{\psi_1\}$
u_5	$p_{51} : \{\psi_1, \psi_2\}$		

$$L = \{\ell_1, \ell_2\}, \quad \Psi = \{\psi_1, \psi_2\}$$
$$\mathcal{U}_{L\Psi} = \{u_1, u_3\}, \quad \mathcal{U}_{L\widetilde{\Psi}} = \{u_1, u_2, u_3\}$$
$$\mathcal{U}_{\Psi} = \{u_1, u_3, u_4, u_5\}, \quad \mathcal{U}_{\widetilde{L}\Psi} = \{u_1, u_3, u_5\}$$
$$sup(L, \Psi) = 2, \quad w\_sup(L, \Psi) = 3, \quad rw\_sup(L, \Psi) = 2$$

Fig. 7.2 Running example.

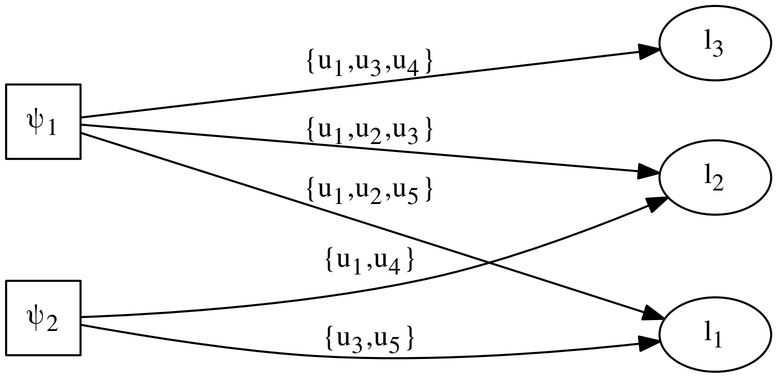


Fig. 7.3 Association Graph for the running example.

An association extracted from a user's posts between a keyword set and a location set could be arbitrary. After all, the content of a post is not always related to the location where it was made, and crowdsourced content is known to be characterized by errors and noise. Hence, an association acquires credence by the number of users supporting it. Accordingly, we use this to measure the strength of a keywords-locations association.

Definition 7.5 (Support)**.** *The* support *of an association between a location set L and keyword set* Ψ *is the number of users supporting* (L, Ψ)*, i.e.,* $sup(L, \Psi) = |\mathcal{U}_{L\Psi}|$.

Returning to our example, user u_1 supports the location set $L = \{\ell_1, \ell_2\}$ and keyword set $\Psi = \{\psi_1, \psi_2\}$. For instance, post p_{11} (resp. p_{12}) is relevant to ψ_1 (resp. ψ_2) and local to some location among L; hence the first condition is satisfied; similarly, the second condition is also satisfied. It is not hard to see that the conditions are also satisfied for user u_3. Therefore, $sup(L, \Psi) = 2$.

We can now formally state the objective of this work. Given a set of keywords, we formulate two variants, one that retrieves all associations above a support threshold, and one that retrieves the k most strongly supported associations.

Problem 7.1 (Frequent Socio-Textual Associations). *Given a keyword set Ψ and a support threshold σ, identify all the location sets, up to cardinality m, that have support above σ.*

Problem 7.2 (Top-k Socio-Textual Associations). *Given a keyword set Ψ, identify k location sets, up to cardinality m, that have the highest support.*

The restriction on the cardinality of the location set is because, as explained in Section 7.4, adding more locations can increase the support of the set.

7.4 Observations and Approach

Our approach is based on some key observations regarding the intrinsic characteristics of the studied problems. In fact, the stated problems reminisce the frequent itemset problem; however, the key difference here is that the introduced support function does not have the necessary anti-monotonicity property which allows for applying the Apriori principle. Given two sets X, Y, this property states that if $X \subseteq Y$, then $sup(X) \geq sup(Y)$. In other words, adding more items to a set cannot increase its support. However, the support introduced in Definition 7.5 does not exhibit this property.

Theorem 7.1. *The support of a location set L and a keyword set Ψ is not anti-monotonic with respect to the location set, i.e., there exist two location sets $L \subseteq L'$ and a keyword set Ψ, such that $sup(L, \Psi) < sup(L', \Psi)$.*

Proof. We prove via an example. Assume three keywords, four locations, and two users who have made posts in exactly those locations, as shown below:

	ℓ_1	ℓ_2	ℓ_3	ℓ_4
u_1	ψ_1	ψ_2	ψ_3	ψ_1
u_2	ψ_3	ψ_1	ψ_1	ψ_2

Consider the keyword set $\Psi = \{\psi_1, \psi_2, \psi_3\}$. Notice that only user u_1 supports location set $L = \{\ell_1, \ell_2, \ell_3\}$, i.e., $sup(L, \Psi) = 1$. On the other hand, both users support location set $L' = \{\ell_1, \ell_2, \ell_3, \ell_4\}$, i.e., $sup(L', \Psi) = 2$. In fact, any 3-location set in this example has support at most 1. □

As a matter of fact, the support of a location set and a keyword set can increase or decrease with respect to the location set. Despite this negative result, we devise an efficient filter-and-refine approach, where the filtering step exploits a weaker support measure.

Definition 7.6 (Weakly Supporting User). *A user u* weakly supports *a given location set L and keyword set Ψ, denoted as $u \in \mathcal{U}_{L\widetilde{\Psi}}$, if for each location $\ell \in L$, the user has made a post local to ℓ and relevant to a keyword in Ψ.*

The difference with respect to Definition 7.4 is that only the second condition applies. In other words, in the Association Graph, there must exist edges associating each one of the locations in L with keywords from Ψ, but without necessarily involving all keywords in Ψ. Accordingly, we define the notion of weak support.

Definition 7.7 (Weak Support). *The* weak support *of a given location set L and keyword set Ψ is the number of users weakly supporting (L, Ψ), i.e., $w\_sup(L, \Psi) = |\mathcal{U}_{L\widetilde{\Psi}}|$.*

In our example, user u_2 weakly supports (L, Ψ), where $L = \{\ell_1, \ell_2\}$ and $\Psi = \{\psi_1, \psi_2\}$. For both locations, u_2 has local posts (p_{21} and p_{22}) that are relevant to at least one keyword (ψ_1). In addition, users u_1, u_3 also weakly support the same location set and keyword set. On the other hand, u_4 and u_5 do not, as they do not have posts local to both locations. Therefore, $w\_sup(L, \Psi) = 3$.

Our filter-and-refine approach hinges on two properties of the weak support. The first is its anti-monotonicity, while the second is that it provides an upper bound for the support of an association.

Lemma 7.1. *The weak support of a location set and a keyword set is anti-monotonic with respect to the location set, i.e., for any two location sets $L' \subseteq L$ and keyword set Ψ, it holds that $w\_sup(L', \Psi) \geq w\_sup(L, \Psi)$.*

Proof. We show that any user u that does not weakly support (L', Ψ) cannot weakly support (L, Ψ). Assume otherwise, meaning that for each location in L there exists a post of u that is local to that location and relevant to the set Ψ. Trivially, this property also holds for any location in $L' \subseteq L$. Therefore, u must also support (L', Ψ) — a contradiction. □

Lemma 7.2. *The support of location set L and keyword set Ψ is not greater than their weak support, i.e., $sup(L,\Psi) \leq w\_sup(L,\Psi)$.*

Proof. We show that any user u that supports (L,Ψ) also weakly supports (L,Ψ). As per Definition 7.4, u has made a post local to each location in L and relevant to a keyword in Ψ (second condition). Therefore, the condition of Definition 7.6 applies and u must also weakly support (L,Ψ). □

Returning to the example, users u_1, u_2, u_3, u_5 weakly support (L',Ψ), where $L' = \{\ell_1\}$. Hence, as per Lemma 7.1, $w\_sup(L',\Psi) \geq w\_sup(L,\Psi)$. Moreover, as per Lemma 7.2, we have seen that the weak support of (L,Ψ) is one more than its support. Based on these lemmas, we can derive the following important property.

Theorem 7.2. *If the weak support of a location set L and a keyword set Ψ is less than σ, then the support of any location set $L' \supseteq L$ and Ψ cannot be more than σ.*

Proof. The premise suggests that $\sigma > w\_sup(L,\Psi)$. From Lemma 7.1 we have that $w\_sup(L,\Psi) \geq w\_sup(L',\Psi)$, while from Lemma 7.2 we get $w\_sup(L',\Psi) \geq sup(L',\Psi)$. Putting all three inequalities together we get $\sigma > sup(L',\Psi)$, i.e., the antecedent. □

This result leads us to the following filter-and-refine strategy. Similar to the candidate generation step of the Apriori algorithm, location sets of increasing cardinality are constructed. Then, the weak support of the set is counted, and if this is below the threshold, the set is filtered out. At the end of entire process (when set cardinality reaches m), the refinement step is performed by explicitly counting the support of all surviving location sets.

Still, this approach could be inefficient, producing many false positives. It is possible that the support of a location set is below the threshold even though its weak support is above the threshold. Its support may even be zero if there exists no user that has posts covering all keywords. Such a location set cannot be pruned by Theorem 7.2. Following our example, consider location set $L = \{\ell_1, \ell_2\}$, keyword set $\Psi = \{\psi_1, \psi_2\}$, and assume that only user u_2 exists. In this case, $w\_sup(L,\Psi) = 1$, but $sup(L,\Psi) = 0$, since there exists no post from u_2 relevant to ψ_2. Motivated by this, we seek additional ways to identify location sets that cannot have high support. We first define the notion of a relevant user.

Definition 7.8 (Relevant User). *We say that a user u is* relevant *to a given keyword set Ψ, and denote as $u \in \mathcal{U}_\Psi$, if for each keyword $\psi \in \Psi$, the user has made a post relevant*

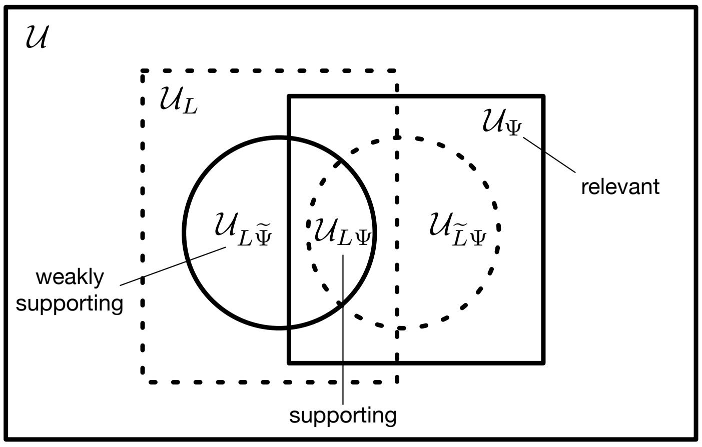


Fig. 7.4 Set relationships between supporting, weakly supporting, and relevant users for the association between location set L and keyword set Ψ.

to ψ, i.e., the Association Graph contains an edge that is adjacent to ψ and includes u in its label.

Notice that user u_2 is not relevant to $\Psi = \{\psi_1, \psi_2\}$. The next result shows that if we restrict the set of weakly supporting users to include only relevant users, we can still define a pruning rule.

Theorem 7.3. *If the number of relevant users that weakly support a location set L and a keyword set Ψ is less than σ, then the support of any location set $L' \supseteq L$ and Ψ cannot be more than σ.*

Proof. Recall that $\mathcal{U}_\Psi$, $\mathcal{U}_{L\widetilde{\Psi}}$ denote the set of relevant users and weakly supporting users, respectively. Then, the theorem assumes that $|\mathcal{U}_\Psi \cap \mathcal{U}_{L\widetilde{\Psi}}| < \sigma$. From (the proof of) Lemma 7.1, we have that $\mathcal{U}_{L\widetilde{\Psi}} \supseteq \mathcal{U}_{L'\widetilde{\Psi}}$. Therefore, $\mathcal{U}_\Psi \cap \mathcal{U}_{L\widetilde{\Psi}} \supseteq \mathcal{U}_\Psi \cap \mathcal{U}_{L'\widetilde{\Psi}}$, and thus $|\mathcal{U}_\Psi \cap \mathcal{U}_{L\widetilde{\Psi}}| \geq |\mathcal{U}_\Psi \cap \mathcal{U}_{L'\widetilde{\Psi}}|$. From (the proof of) Lemma 7.2, any user u that supports (L', Ψ) must also weakly support (L', Ψ). In addition, u must be relevant to Ψ due to the first condition of Definition 7.4. Hence, $|\mathcal{U}_\Psi \cap \mathcal{U}_{L'\widetilde{\Psi}}| \geq sup(L', \Psi)$. Combining the two derived inequalities and the theorem assumption, we derive that $sup(L', \Psi) < \sigma$. □

This result improves upon our filter-and-refine strategy, by allowing us to early prune a location set that cannot have support above σ, even though its weak support might be above σ.

A better way to understand the relation between the sets of supporting $\mathcal{U}_{L\Psi}$, weakly supporting $\mathcal{U}_{L\widetilde{\Psi}}$, and relevant $\mathcal{U}_\Psi$ users of a location set and keyword set (L, Ψ) is to draw a Venn diagram. Figure 7.4 depicts these sets and also includes for completeness their dual sets drawn with dashed lines (discussed in Section 7.5.2). We have shown that while the cardinality of set $\mathcal{U}_{L\Psi}$ is not anti-monotone with respect to L, the cardinalities of sets $\mathcal{U}_{L\widetilde{\Psi}}$ and $\mathcal{U}_{\Psi} \cap \mathcal{U}_{L\widetilde{\Psi}}$ are. Figure 7.4 emphasizes that the intersection of relevant and weakly supporting users is a *tighter* superset of the desired supporting users set, while still allowing anti-monotonicity-based pruning. In the following, we write $rw\_sup(L, \Psi)$ to denote the number of relevant and weakly supporting users, i.e., $|\mathcal{U}_{\Psi} \cap \mathcal{U}_{L\widetilde{\Psi}}|$.

Returning to the example of Figure 7.2, the relevant to Ψ users are all except u_2. Therefore, we derive $sup(L, \Psi) = |\{u_1, u_3\}| = 2$, $w\_sup(L, \Psi) = |\{u_1, u_2, u_3\}| = 3$, and $rw\_sup(L, \Psi) = |\{u_1, u_3\}| = 2$, showing that the relevant and weak support is closer to the actual support than weak support is.

7.5 Finding Frequent Associations

We first present a baseline method for Problem 7.1, which serves as the foundation for more elaborate solutions based on indexes.

7.5.1 Basic Algorithm

This algorithm implements the filter-and-refine approach discussed in Section 7.4. Recall that Theorems 7.2 and 7.3 allow to prune location sets with support less than σ based on the concepts of relevant and weakly supporting users (filter step). While this guarantees no false negatives, there can still be false positives, i.e., location sets with support less than σ, which need to be identified (refine step). Note that instead of performing this at the end, it can be done more efficiently during candidate generation, as explained later.

Algorithm 7.1 outlines the basic method, denoted as STA. It operates on the set $\mathcal{P}$ of posts organized by user, i.e., the list $\mathcal{P}_u$ containing the posts of each user u. The input includes the keyword set Ψ, the maximum cardinality m of a location set, and the support threshold σ. STA exploits the Apriori principle (lines 4–12) to identify the location sets with support above σ, filtering out each location set with fewer than σ relevant and weakly supporting users.

Initially, the result set is empty and the potential 1-location sets are set to all locations (lines 1–2). Also, the set of users relevant to Ψ is identified (line 3). Procedure IDENTIFYRELEVANTUSERS, depicted in Algorithm 7.2, iterates across every list $\mathcal{P}_u$ and checks if user u has made posts that cover all keywords that appear in Ψ.

Algorithm 7.1: Algorithm STA

```
Input: keyword set $\Psi$, maximum cardinality $m$, support threshold $\sigma$
Output: result set $\mathcal{R}^\sigma$ of all location sets with support at least $\sigma$
1  $\mathcal{R}^\sigma \leftarrow \emptyset$
2  $\mathcal{C}_1 \leftarrow \mathcal{L}$                                    ▷ candidate 1-location sets
3  $\mathcal{U}_\Psi \leftarrow$ IdentifyRelevantUsers($\Psi$)
4  for $1 \leq i \leq m$ do
5      $\mathcal{F}_i \leftarrow \emptyset$   ▷ $i$-location sets with more than $\sigma$ relevant and weakly supporting users
6      foreach $L \in \mathcal{C}_i$ do
7          ComputeSupports($L$, $\Psi$)
8          if $rw\_sup(L, \Psi) \geq \sigma$ then
9              $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \{L\}$
10             if $sup(L, \Psi) \geq \sigma$ then
11                 $\mathcal{R}^\sigma \leftarrow \mathcal{R}^\sigma \cup \{L\}$
12     $\mathcal{C}_{i+1} \leftarrow$ CandidateGeneration($\mathcal{F}_i$)          ▷ candidate $(i+1)$-location sets
```

Then, STA proceeds in m iterations, following the Apriori principle. At the i-th iteration, all i-location sets with rw_sup not less than σ are stored in set $\mathcal{F}_i$. Among them, those with support not less than σ are added to the result set $\mathcal{R}^\sigma$. After initializing $\mathcal{F}_i$ (line 5), each candidate i-location set L is examined (lines 6–11). The set $\mathcal{C}_i$ of candidate i-location sets was generated at the end of the previous iteration (line 12) by the CandidateGeneration procedure that applies the Apriori principle. In particular, Candidate Generation creates candidate location sets of cardinality one more than what was just examined. It takes as input the i-location sets $\mathcal{F}_i$ with relevant weak support above σ and inserts into $\mathcal{C}_{i+1}$ an $(i+1)$-location set only if all its i-location subsets are in $\mathcal{F}_i$, due to the Apriori principle implied by Theorem 7.3.

For candidate i-location set L, procedure ComputeSupports (described later) is invoked to determine the number $rw\_sup(L, \Psi)$ of relevant weakly supporting users, and the number $sup(L, \Psi)$ of supporting users (line 7). If the former support is above σ, L is added to $\mathcal{F}_i$ (lines 8–9). If, additionally, the latter support is greater than σ, then L is added to the result set $\mathcal{R}^\sigma$ (lines 10–11). This essentially corresponds to refining the surviving candidates.

Algorithm 7.3 depicts the pseudocode for ComputeSupports. The procedure iterates over all relevant users. Let u be the currently examined user. The objective is to determine if u (weakly) supports (L, Ψ). For this purpose, the sets $covL$ and $cov\Psi$ are constructed to indicate what locations among L and keywords among Ψ, respectively, are covered by u. Each post of u is examined (lines 4–9). If the post's

Algorithm 7.2: STA.IdentifyRelevantUsers

```
Input: keyword set $\Psi$
Output: set $\mathcal{U}_\Psi$ of relevant users
1 $\mathcal{U}_\Psi \leftarrow \emptyset$
2 foreach $u \in \mathcal{U}$ do
3     $cov\Psi \leftarrow \emptyset$
4     foreach $p \in \mathcal{P}_u$ do
5         if $p.\psi \in \Psi$ then
6             $cov\Psi \leftarrow cov\Psi \cup \{\psi\}$
7     if $|cov\Psi| = |\Psi|$ then
8         $\mathcal{U}_\Psi \leftarrow \mathcal{U}_\Psi \cup \{u\}$
```

location is within distance ϵ to some location in $\ell \in L$, and there exists a keyword $\psi \in \Psi$ common with the post's keywords, then ℓ and ψ are inserted to $covL$ and $cov\Psi$ (lines 6–9). If all keywords in L have been found in u's relevant posts, then the counter of relevant and weakly supporting users is incremented (lines 10–11). Additionally, if all keywords appear in these posts, the counter for the support is incremented (lines 12–13).

Table 7.3 shows the relevant and weak support, and support for all location sets for keyword set $\Psi = \{\psi_1, \psi_2\}$, as computed by STA for the example of Figure 7.2 with support threshold $\sigma = 2$. Recall that all users except u_2 are relevant. As all 1-location sets have relevant and weak support above σ (although none is actually a result), all possible 2-location sets are constructed and their supports are counted. Among them, $\{\ell_1, \ell_2\}$ and $\{\ell_2, \ell_3\}$ (marked bold) have support 2 and are thus results. Observe the anti-monotonicity in relevant and weak support, and the lack thereof in support. Finally, as all 2-location sets have wr_sup above σ, the set $\{\ell_1, \ell_2, \ell_3\}$ is also considered but found to have low relevant and weak support.

7.5.2 Inverted Index-Based Algorithm

In STA, counting the weak support of a location set is particularly time consuming, since it scans the entire list of posts to find the weakly supporting users for each location. Even worse, if a location is part of multiple location sets, this is repeated multiple times.

To address this performance bottleneck, we present next an approach, termed STA-I, that is based on a preconstructed inverted index, which facilitates the identifi-

Algorithm 7.3: STA.ComputeSupports

```
Input: location set $L$, keyword set $\Psi$
Output: weak support and support of $(L, \Psi)$
1  $r\_sup(L,\Psi) \leftarrow 0$; $sup(L,\Psi) \leftarrow 0$
2  foreach $u \in \mathcal{U}_\Psi$ do                                ▷ relevant user
3      $covL \leftarrow \emptyset$; $cov\Psi \leftarrow \emptyset$
4      foreach $p \in \mathcal{P}_u$ do
5          foreach $\ell \in L$ do
6              if $d(p.\ell, \ell) \leq \epsilon$ then
7                  foreach $\psi \in p.\Psi \cap \Psi$ do
8                      $covL \leftarrow covL \cup \{\ell\}$
9                      $cov\Psi \leftarrow cov\Psi \cup \{\psi\}$
10     if $|covL| = |L|$ then                                ▷ weakly supporting user
11         $rw\_sup(L,\Psi) \leftarrow rw\_sup(L,\Psi) + 1$
12         if $|cov\Psi| = |\Psi|$ then                        ▷ supporting user
13             $sup(L,\Psi) \leftarrow sup(L,\Psi) + 1$
```

Algorithm 7.4: STA-I.IdentifyRelevantUsers

```
Input: keyword set $\Psi$
Output: set $\mathcal{U}_\Psi$ of relevant users
1  $\mathcal{U}_\Psi \leftarrow \emptyset$
2  foreach $\psi \in \Psi$ do
3      $\mathcal{C} \leftarrow \emptyset$
4      foreach $\ell \in \mathcal{L}$ do
5          $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{U}(\ell, \psi)$
6      $\mathcal{U}_\Psi \leftarrow \mathcal{U}_\Psi \cap \mathcal{C}$
```

cation of weakly supporting users for any location. The assumption here is that the distance parameter ϵ is known beforehand, i.e., it does not change with the queries.

To construct the index, we identify the posts that are within distance ϵ to each location ℓ. Then, for each location, we compile an inverted list $\mathcal{U}(\ell)$, containing all users with posts local to ℓ. To further speed up processing, we partition each list according to keyword, such that each sublist $\mathcal{U}(\ell, \psi)$ contains users with posts local to ℓ and relevant to ψ. Table 7.4 shows the lists for our example. STA-I operates identically to STA, but uses the inverted index during the procedures IdentifyRelevantUsers and ComputeSupports.

The IdentifyRelevantUsers procedure is depicted in Algorithm 7.4. Recall that the goal is to identify users who have made posts relevant to all keywords in Ψ,

Table 7.3 Support of associations between listed location sets and keyword set $\Psi = \{\psi_1, \psi_2\}$ based on the posts in Figure 7.2.

Location set	**wr_sup**	**sup**
$\{\ell_1\}$	3	1
$\{\ell_2\}$	3	1
$\{\ell_3\}$	3	0
$\boldsymbol{\{\ell_1, \ell_2\}}$	2	**2**
$\{\ell_1, \ell_3\}$	2	1
$\boldsymbol{\{\ell_2, \ell_3\}}$	3	**2**
$\{\ell_1, \ell_2, \ell_3\}$	1	1

Table 7.4 Inverted index for the posts in Figure 7.2.

Location	**Inverted list**
ℓ_1	$\psi_1 : u_1, u_5,\ \ \psi_2 : u_3, u_5$
ℓ_2	$\psi_1 : u_1, u_3,\ \ \psi_2 : u_1, u_4$
ℓ_3	$\psi_1 : u_1, u_3, u_4$

irrespective of the posts' geotags. Hence, for each keyword $\psi \in \Psi$ and each possible location ℓ, it retrieves the list $\mathcal{U}(\ell, \psi)$ of users with relevant and local posts, and it compiles the set of users with posts relevant to ψ and local to some location in L. Finally, it computes the intersection of these sets. This procedure essentially constructs the set of relevant users as $\mathcal{U}_\Psi = \bigcap_{\psi \in \Psi} \left(\bigcup_{\ell \in L} \mathcal{U}(\ell, \psi)\right)$.

Algorithm 7.5 illustrates the ComputeSupports procedure, which computes the weak support (lines 1–6) and the support (lines 8–14) of location set and keyword set (L, Ψ). Regarding the former, recall that a user weakly supports (L, Ψ) if for each location $\ell \in L$ there exists a local post that is relevant to some keyword in Ψ. The set $\bigcup_{\psi \in \Psi} \mathcal{U}(\ell, \psi)$ represents users that have relevant posts to some keyword in Ψ and are local to the specific location ℓ. Thus, the intersection over all locations in L of these sets represents the weakly supporting users, i.e., $\mathcal{U}_{L\widetilde{\Psi}} = \bigcap_{\ell \in L} \left(\bigcup_{\psi \in \Psi} \mathcal{U}(\ell, \psi)\right)$. Specifically, the procedure computes the union in the inner loop (lines 3–4) and the intersection of the unions in the outer loop (lines 2–5). The weak support of (L, Ψ) is computed after the non-relevant users are discarded (line 6).

Only when the weak support of (L, Ψ) exceeds threshold σ (line 7), its support is computed (lines 8–14), but in a manner significantly different from that in STA. Recall from the discussion in Section 7.4 and Figure 7.4 that the set $\mathcal{U}_{L\widetilde{\Psi}}$ of weakly supporting users has a dual set $\mathcal{U}_{\widetilde{L}\Psi}$, termed local-weakly supporting users. This

Algorithm 7.5: STA-I.ComputeSupports

```
   Input: location set $L$, keyword set $\Psi$
   Output: weak support and support of $(L, \Psi)$
   ▷ construct set $\mathcal{U}_{L\widetilde{\Psi}}$ of (relevant-)weakly supporting users
1  $\mathcal{U}_{L\widetilde{\Psi}} \leftarrow \emptyset$
2  foreach $\ell \in L$ do
3      $\mathcal{A} \leftarrow \emptyset$ foreach $\psi \in \Psi$ do
4          $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{U}(\ell, \psi)$
5      $\mathcal{U}_{L\widetilde{\Psi}} \leftarrow \mathcal{U}_{L\widetilde{\Psi}} \cap \mathcal{A}$
6  $rw\_sup(L, \Psi) \leftarrow |\mathcal{U}_{L\widetilde{\Psi}} \cap \mathcal{U}_{\Psi}|$
7  if $rw\_sup(L, \Psi) < \sigma$ then return
   ▷ construct set $\mathcal{U}_{\widetilde{L}\Psi}$ of local-weakly supporting users
8  $\mathcal{U}_{\widetilde{L}\Psi} \leftarrow \emptyset$
9  foreach $\psi \in \Psi$ do
10     $\mathcal{B} \leftarrow \emptyset$
11     foreach $\ell \in L$ do
12         $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{U}(\ell, \psi)$
13     $\mathcal{U}_{\widetilde{L}\Psi} \leftarrow \mathcal{U}_{\widetilde{L}\Psi} \cap \mathcal{B}$
14 $sup(L, \Psi) \leftarrow |\mathcal{U}_{L\widetilde{\Psi}} \cap \mathcal{U}_{\widetilde{L}\Psi}|$
```

latter set contains users that for each keyword among Ψ have a post local to some location among L. It is not hard to see that the users that are both (relevant-)weakly supporting and local-weakly supporting (L, Ψ) are exactly those that support (L, Ψ), i.e., it holds that $\mathcal{U}_{L\Psi} = \mathcal{U}_{L\widetilde{\Psi}} \cap \mathcal{U}_{\widetilde{L}\Psi}$. Intuitively, the latter set satisfies the first requirement of Definition 7.4, whereas the former the second.

Based on this observation, the ComputeSupports procedure first computes the local-weakly supporting users $\mathcal{U}_{\widetilde{L}\Psi}$ (lines 8–13). With similar reasoning as before, the procedure builds the set as $\mathcal{U}_{\widetilde{L}\Psi} = \bigcap_{\psi \in \Psi} (\bigcup_{\ell \in L} \mathcal{U}(\ell, \psi))$, where the union is compiled in the inner loop (lines 11–12) and the intersection of the unions in the outer loop (lines 9–13). Then, it intersects it with the previously constructed $\mathcal{U}_{L\widetilde{\Psi}}$ set to compute the support of (L, Ψ) (line 14).

7.5.3 Spatio-Textual Index-Based Algorithm

Although precomputing the inverted index reduces dramatically the cost of calculating the weak support of a location set, it cannot handle different values of the distance parameter ϵ. Next, we present an alternative approach to accelerating weak support calculations based on spatio-textual indexes. Instead of relying on precomputed static lists, we dynamically compile the information needed from the index. We first present a generic approach that works with the majority of existing spatio-textual indexes, and then we consider a particular index and propose further optimizations.

Generic Algorithm

We adapt the basic Apriori-like algorithm assuming the availability of a spatio-textual index which can process spatio-textual range queries with OR semantics. The latter specify a spatial range R and a set of keywords Ψ, and seek all spatio-textual objects whose location is inside R and contain at least one of the keywords in Ψ.

We next describe the STA-ST algorithm which operates on top of such a general-purpose spatio-textual index. It operates similarly to STA, with the difference that procedure ComputeSupports is implemented in an index-aware manner, as outlined in Algorithm 7.6. It first constructs the set $\mathcal{U}_{L\widetilde{\Psi}}$ of weakly supporting users, and then determines the support of (L,Ψ). To build $\mathcal{U}_{L\widetilde{\Psi}}$, it issues a spatio-textual range query with parameters the disc (ℓ,ϵ) of radius ϵ around each location $\ell \in L$ and keyword set Ψ (lines 2–9). For a specific location ℓ, the results (set of posts) are stored in $\mathcal{P}_\ell$ (line 4). Then, it scans the results and inserts into a temporary variable $\mathcal{A}$ each encountered user $p.u$ (line 8). In addition, it associates with each user a bitmap $p.u.cov\Psi$ indicating which query keywords appear in her posts (lines 6–7); this information is later used to determine if the user supports (L,Ψ). Once all users with posts local to ℓ and relevant to Ψ have been identified in $\mathcal{A}$, they are merged with the ones for previously examined locations (line 9). Eventually, $\mathcal{U}_{L\widetilde{\Psi}}$ contains users with posts local to *every* location in L and relevant to at least one keyword in Ψ, i.e., the users weakly supporting (L,Ψ).

To compute the weak support among relevant users, the procedure takes the intersection of $\mathcal{U}_{L\widetilde{\Psi}}$ with the known set $\mathcal{U}_\Psi$ of relevant users (line 10). If the weak support is lower than the threshold, the algorithm returns (line 11). Otherwise it computes the support by examining whether each user has covered all query keywords (lines 13–15); this is determined directly from bitmaps $p.u.cov\Psi$.

Optimized Algorithm

Next, we focus on a specific spatio-textual index, I^3 [175], which we adapt to devise an even more efficient algorithm.

For our purposes, the I^3 index can be seen as a quadtree that hierarchically partitions the spatial domain. Each node corresponds to a specific rectangular region and points to its four children corresponding to the quadrants of the region. Leaf

Algorithm 7.6: STA-ST.ComputeSupports

```
Input: location set $L$, keyword set $\Psi$
Output: weak support and support of $(L,\Psi)$
$\mathcal{U}_{L\widetilde{\Psi}} \leftarrow \emptyset$
foreach $\ell \in L$ do
    $\mathcal{A} \leftarrow \emptyset$
    $\mathcal{P}_\ell \leftarrow$ ST-RANGE($(\ell,\epsilon),\Psi$)
    foreach $p \in \mathcal{P}_\ell$ do
        foreach $\psi \in p.\Psi \cap \Psi$ do
            $p.u.cov\Psi \leftarrow p.u.cov\Psi \cup \{\psi\}$
        $\mathcal{A} \leftarrow \mathcal{A} \cup p.u$
    $\mathcal{U}_{L\widetilde{\Psi}} \leftarrow \mathcal{U}_{L\widetilde{\Psi}} \cap \mathcal{A}$
$rw\_sup(L,\Psi) \leftarrow |\mathcal{U}_{L\widetilde{\Psi}} \cap \mathcal{U}_\psi|$
if $rw\_sup(L,\Psi) < \sigma$ then return
$sup(L,\Psi) \leftarrow 0$
foreach $u \in \mathcal{U}_{L\widetilde{\Psi}}$ do
    if $|u.cov\Psi| = |\Psi|$ then
        $sup(L,\Psi) \leftarrow sup(L,\Psi) + 1$
```

nodes point to disk pages containing the actual posts grouped by keyword. We associate with each node some additional aggregate information. Specifically, for each keyword ψ, we store the number of users with relevant posts that are contained within the sub-tree rooted at this node N. We denote this by $N.count(\psi)$.

STA-STO differs from STA-ST in the first iteration of the main Apriori loop (lines 4–12 of Algorithm 7.1 for $i = 1$). Instead of computing the weak support (and support) of every location, it uses the index to identify locations with potentially high weak support, eliminating groups of locations with weak support less than σ. To achieve this, it executes a best-first search (bfs) traversal [86], performing a simple test at each node to decide whether to continue in its sub-tree. Intuitively, we wish to terminate bfs when no location in the sub-tree can have weak support greater than σ.

Let Q be the priority queue implementing bfs. For each node N entering Q, the algorithm computes $a(N) = \sum_{\psi\in\Psi} N.count(\psi)$, and uses it as the queue's priority key. At each iteration, the node N in Q with the largest $a(N)$ value is removed. If $a(N)$ is greater than or equal to σ, there may exist some location in the sub-tree of N with weak support greater than σ. Otherwise, a safe conclusion cannot be drawn. Hence, the algorithm calculates an additional value $b(N)$ for this node, which is an upper bound on the weak support of any location within N. Clearly, if $b(N) < \sigma$, the node contains no useful locations and can be pruned. Such pruned nodes, along with their $a()$ values, are maintained in a deleted list D, which serves in the calculation of $b()$ values as explained next. For node N, its $b(N)$ value is the sum of $a()$ values for all nodes that are in Q or in D and that are within distance ϵ to N. An important observation here is that, due to the bfs traversal and the index structure, nodes in $Q \cup D$ do not spatially overlap and hence $b(N)$ does not double count posts. To summarize, STA-STO first makes the quick $a(N) \geq \sigma$ test, and only if this fails does it compute $b(N)$ and makes the more expensive $b(N) \geq \sigma$ test. If the latter fails too, the node definitely cannot contain a location set with weak support greater than σ.

For each location dequeued in the bfs traversal, STA-STO invokes the procedure STA-ST.ComputeSupport as described in the previous section, to determine its exact weak support and its support. Compared to it, the benefit is that STA-STO executes the procedure only for promising locations instead of every possible location.

7.6 Finding Top-k Associations

Next, we present algorithms for Problem 7.2. We start with a basic approach, and then discuss more efficient index-based techniques.

7.6.1 Basic Algorithm

In Problem 7.2, we seek the top-k location sets with the highest support, instead of setting a specific support threshold. However, a support threshold is needed in order to apply an Apriori-like method; thus, we explain how such a threshold can be computed. If we pick any set of k distinct location sets and compute their supports, then the minimum value among those can serve as the support threshold σ; clearly, any other set with support lower than this cannot be in the result. The challenge is to construct initial location sets with high support so that the starting value of σ is effectively high.

Algorithm 7.7 outlines the generic method K-STA implementing this simple idea. First, procedure DetermineSupportThreshold is invoked to obtain an appropriate lower bound σ on the support of the top-k set. Given σ, it invokes the STA algorithm to derive all location sets with support above σ. Finally, among the returned location sets, it returns the k with the highest support.

Regarding the DetermineSupportThreshold procedure, the main idea is to construct at least k distinct location sets that cover all keywords Ψ. Suppose that for each keyword $\psi \in \Psi$ we have determined $k(\psi)$ distinct locations with local posts

Algorithm 7.7: Algorithm K-STA

```
Input: keyword set $\Psi$, maximum cardinality $m$, number of results $k$
Output: result set $\mathcal{R}^k$ containing top-$k$ location sets with highest support
1 $\sigma \leftarrow$ DetermineSupportThreshold($\Psi$, $k$)
2 $\mathcal{R}^\sigma \leftarrow$ STA ($\Psi$, $m$, $\sigma$)
3 $\mathcal{R}^k \leftarrow k$ location sets from $\mathcal{R}^\sigma$ with highest support
```

relevant to ψ. Combining these $k(\psi)$ distinct locations for each keyword, we can construct distinct location sets. Note that a necessary condition to obtain k location sets is $\prod_{\psi \in \Psi} k(\psi) \geq k$.

Following this process, a heuristic for obtaining combinations with high support is to start with locations that are popular, i.e., have high weak support. In the absence of any index, procedure DetermineSupportThreshold iterates over the set of posts lists $\mathcal{P}_u$, skipping users that do not have relevant posts to each ψ. For the rest, the locations of the relevant posts to each ψ are noted. In addition, a counter for the weak support of each location is maintained. After a sufficient number of locations for each keyword are seen, the procedure terminates. For each keyword, the locations with the highest weak support are chosen and combined. The support of each set is computed by ComputeSupports, and the k-th highest among these values is set as the support threshold σ.

7.6.2 Index-Based Algorithms

Inverted Index

When an inverted index from locations to users with local posts is available, the routine DetermineSupportThreshold collects locations with local posts relevant to each keyword in Ψ in a different manner. It first computes the weak support of every location by invoking ComputeSupports. Note that this has to be executed anyway when we later invoke the STA-I algorithm irrespective of the support threshold σ. Then, it examines locations in descending order of their weak support. For each location ℓ, the procedure checks the inverted list and associates the location with each keyword in Ψ for which a local and relevant post exists. Similar to the basic algorithm, once a sufficient number of locations per keyword are seen, location sets are generated and their support is computed.

Table 7.5 Datasets used in the experiments.

Dataset	Number of photos	Number of users	Number of distinct tags	Avg. num. of tags per photo	Avg. num. of tags per user	Number of locations
London	1,129,927	16,171	266,495	8.1	61.2	48,547
Berlin	275,285	7,044	88,783	8.1	39.4	21,427
Paris	549,484	11,776	122,998	7.8	38.8	38,358

Table 7.6 Most popular keywords (10 of 30) used to generate queries.

London	Berlin	Paris
thames (2752)	reichstag (876)	louvre (2287)
park (1738)	fernsehturm (774)	eiffel+tower (1742)
london+eye (1730)	architecture (716)	seine (1488)
big+ben (1698)	alexanderplatz (713)	notre+dame (1244)
westminster (1543)	wall (684)	street (1194)
architecture (1519)	graffiti (575)	montmartre (1184)
museum (1386)	street (562)	architecture (1136)
art (1319)	art (543)	museum (1022)
tower+bridge (1276)	museum (526)	church (980)
statue (1178)	spree (492)	art (970)

Spatio-Textual Index

In a generic spatio-textual index, DETERMINESUPPORTTHRESHOLD operates identically to the basic algorithm with the exception that the COMPUTESUPPORTS procedure is index-aware. When the augmented I^3 index is used, a different process is followed. Procedure DETERMINESUPPORTTHRESHOLD first performs a best-first search traversal similar to that described in Section 15. The difference is that initially there is no support threshold, and thus the $b()$ values need not be computed. Moreover, the traversal is progressive, meaning that at each step the next location with potentially high weak support is identified. For each such location, its local posts are retrieved (using the index) and it is marked for the keywords that appear in these posts. As before, once a sufficient number of locations per keyword are seen, the support threshold is computed.

7.7 Experimental Evaluation

In this section, we present an experimental evaluation of our approach using real-world datasets comprising geolocated Flickr photos. We first describe our experimen-

Table 7.7 Most popular keyword sets (5 of 20) used as queries.

$\|\Psi\|$	**London**
2	london+eye, thames (922); big+ben, london+eye (908); thames, westminster (898); park, thames (880); big+ben, thames (846)
3	big+ben, london+eye, thames (557); big+ben, thames, westminster (497); big+ben, london+eye, westminster (472); london+eye, thames, westminster (464); park, thames, westminster (440)
4	big+ben, london+eye, thames, westminster (358); big+ben, london+eye, thames, tower+bridge (293); art, green, park, thames (258); green, park, thames, trees (257); park, statue, thames, westminster (257)
	Berlin
2	alexanderplatz, fernsehturm (404); fernsehturm, reichstag (320); alexanderplatz, reichstag (253); reichstag, wall (249); fernsehturm, spree (248)
3	alexanderplatz, fernsehturm, reichstag (192); alexanderplatz, fernsehturm, spree (166); alexanderplatz, fernsehturm, wall (145); brandenburger+tor, fernsehturm, reichstag (144); fernsehturm, reichstag, spree (142)
4	alexanderplatz, fernsehturm, reichstag, spree (106); alexanderplatz, brandenburger+tor, fernsehturm, reichstag (96); alexanderplatz, fernsehturm, reichstag, wall (95); alexanderplatz, fernsehturm, potsdamer+platz, reichstag (90); alexanderplatz, fernsehturm, museum, reichstag (82)
	Paris
2	eiffel+tower, louvre (777); louvre, seine (745); louvre, museum (706); louvre, notre+dame (691); eiffel+tower, notre+dame (606)
3	eiffel+tower, louvre, notre+dame (415); eiffel+tower, louvre, seine (343); louvre, notre+dame, seine (339); louvre, river, seine (327); arc+de+triomphe, eiffel+tower, louvre (324)
4	eiffel+tower, louvre, notre+dame, seine (215); bridge, louvre, river, seine (209); arc+de+triomphe, eiffel+tower, louvre, notre+dame (208); louvre, museum, river, seine (189); bridge, river, seine, street (187)

tal setup, outlining the datasets and the queries used in the experiments, and then we report and discuss the results.

7.7.1 Datasets

In our experiments, we have used geolocated photos from Flickr, extracted from the large-scale dataset that is provided publicly by Yahoo! for research purposes [153]. Specifically, we compiled datasets for the cities of London, Berlin, and Paris. For each

Table 7.8 Index construction time and size.

	Inverted Index		**I^3 Index**	
	Time (sec)	*Size (MB)*	*Time (sec)*	*Size (GB)*
London	76	215	226	31
Berlin	18	48	38	9
Paris	43	116	90	23

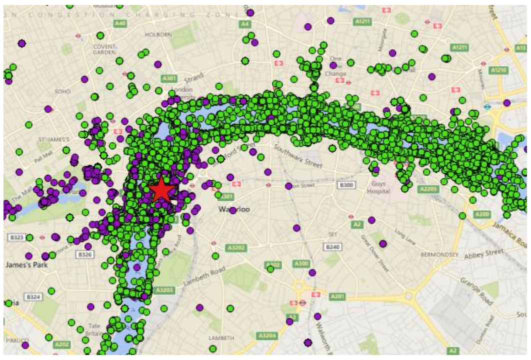
dataset, Table 7.5 lists the number of photos, users, and distinct keywords contained in it, as well as the average number of keywords per photo and distinct keywords per user. As a database of locations, we used POIs collected from the Foursquare API[1]. The number of distinct locations per city is also shown in Table 7.5.

To construct a keyword set that is used to search for socio-textual associations, we followed the process described next. First, for each dataset, we retrieved the 100 most frequent keywords, where the frequency of a keyword was measured by the number of users having photos with it. From those, we manually picked a set of 30 keywords, removing more generic ones, such as *"london"*, *"england"*, *"uk"*, *"iphone"*, *"canon"*, etc. The top 10 selected keywords for each city are listed in Table 7.6, showing also the number of users with relevant posts to each one. Then, we combined these popular keywords to create keyword sets of cardinality up to 4. For each case, we selected the top 20 combinations according to the number of users having photos with those tags. Table 7.7 lists the first 5 among these 20 combinations for each case.

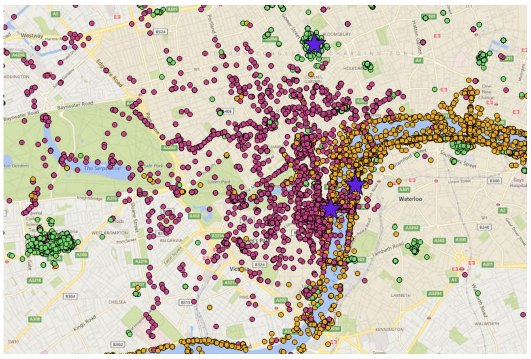
All algorithms were implemented in Java and the experiments were conducted on a machine with Intel® Core™i7-5600U CPU @ 2.60GHz Processor and 16 GB RAM. In all reported experiments, we set the value of the spatial distance threshold parameter ϵ, used to associate photos to locations, to 100 meters.

For the two indexes used by our algorithms, i.e., the inverted index and the (augmented) I^3 index, Table 7.8 shows the index construction time and size for each dataset.

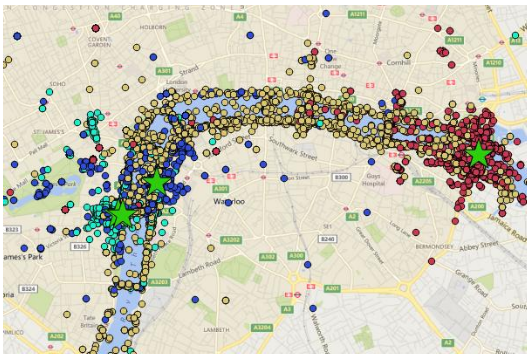
[1]https://developer.foursquare.com/



(a) $\Psi = \{$*"london eye"*, *"thames"*$\}$



(b) $\Psi = \{$*"museum"*, *"thames"*, *"westminster"*$\}$



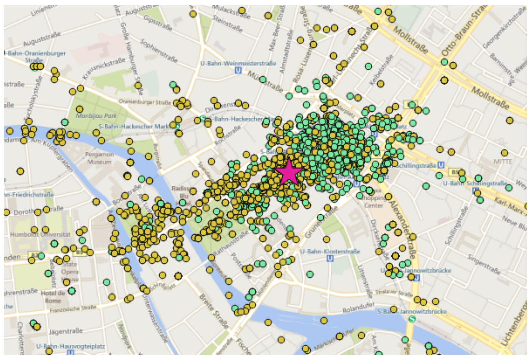
(c) $\Psi = \{$*"big ben"*, *"london eye"*, *"thames"*, *"tower bridge"*$\}$

Fig. 7.5 Sample results for London.

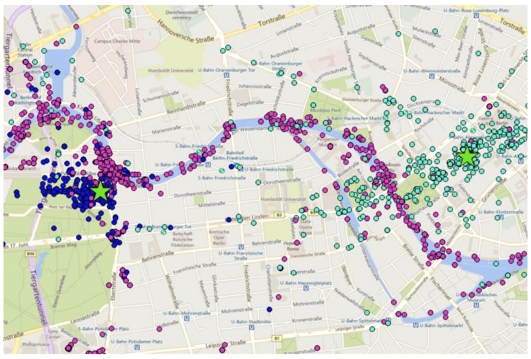
7.7.2 Result Characteristics

We first inspect the top results for a sample of queries in order to assess the results from a qualitative perspective. Specifically, for each of the three datasets, we have selected a sample of three queries with different cardinalities, and we present the top result for each query. The results are presented in Figures 7.5, 7.6, and 7.7. Specifically, the results displayed in each figure are as follows. First, for each keyword in the corresponding query, we retrieve the list of users having photos with that keyword and we intersect these lists to obtain a list of users having photos with all the query keywords. Then, we display the locations of those photos on the map, using different colors for each keyword. Finally, the location(s) contained in the top location set returned by our method are displayed with a star.

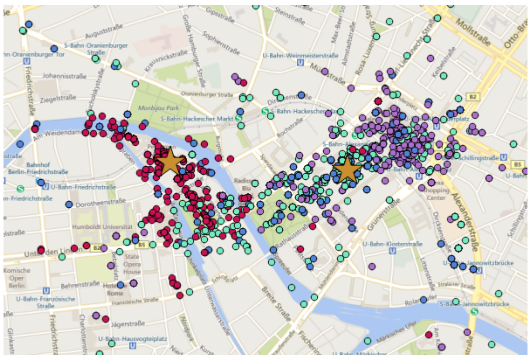
For example, Figure 7.5(a) illustrates the results for the query with keyword set $\Psi = \{$*"london eye"*, *"thames"*$\}$. In this case, the green (resp., purple) points denote the locations of photos that contain the tag *"thames"* (resp., *"london eye"*) and belong to a user that has also posted photos containing the tag *"london eye"* (resp., *"thames"*). We can see that photos about *"thames"* are spread across the whole length of the river Thames.



(a) Ψ = {*"alexanderplatz"*, *"fernsehturm"*}



(b) Ψ = {*"fernsehturm"*, *"reichstag"*, *"spree"*}

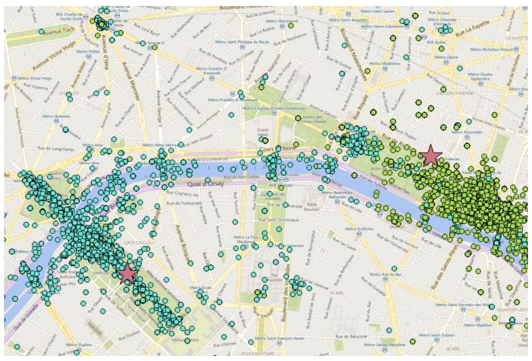


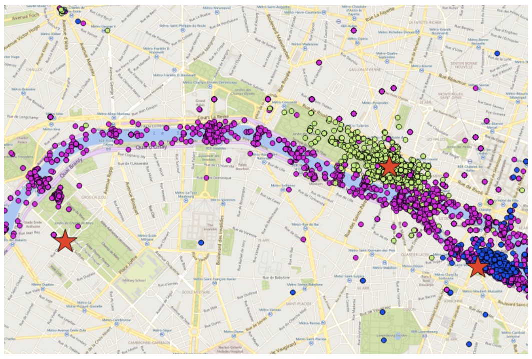
(c) Ψ = {*"alexanderplatz"*, *"fernsehturm"*, *"museum"*, *"tower"*}

Fig. 7.6 Sample results for Berlin.

On the other hand, London Eye is a landmark having a specific location; nevertheless, due to its high visibility, relevant photos can be found at various other locations, especially in and around St. James Park, for example. Moreover, as it happens, in this case where London Eye is located at the banks of river Thames, the regions covered by the respective sets of relevant photos have a high overlap. In fact, the location set found to have the highest support for this query comprises a single location, which, as depicted in the figure, is situated in an area where a large number of photos containing both tags exist. Interestingly, this type of result can also be observed in other examples, where a similar spatial relationship occurs among the relevant entities. For instance, for the query {*"alexanderplatz"*, *"fernsehturm"*} in Berlin, where the Berlin TV Tower is located close to the Alexanderplatz square, the top result comprises a single location.

On the other hand, for the query {*"museum"*, *"thames"*, *"westminster"*} illustrated in Figure 7.5(b), two nearby but distinct locations are included in the top result corresponding to the river Thames and the Westminster Abbey. With respect to the keyword *"museum"*, we can observe in the figure that there exist (at least) two prominent regions with high density of relevant photos, namely one around the British Museum and one around the Natural History Museum and the Victoria and Albert museum. The former has been selected in the top result, indicating that this combination occurs more frequently.


(a) Ψ = {*"eiffel tower"*, *"louvre"*}


(b) Ψ = {*"louvre"*, *"notre dame"*, *"seine"*}

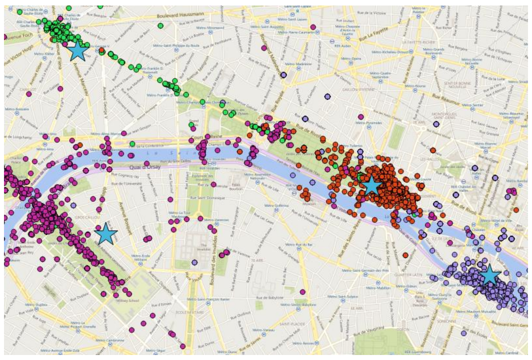

(c) Ψ = {*"arc de triomphe"*, *"eiffel tower"*, *"louvre"*, *"notre dame"*}

Fig. 7.7 Sample results for Paris.

Similarly intuitive results can be observed for the rest of the queries, such as the two locations selected to cover the keyword set {*"fernsehturm"*, *"reichstag"*, *"spree"*} in Berlin (Figure 7.6(b)) or those for {*"eiffel tower"*, *"louvre"*} in Paris (Figure 7.7(a)).

7.7.3 Comparison with Other Association Types

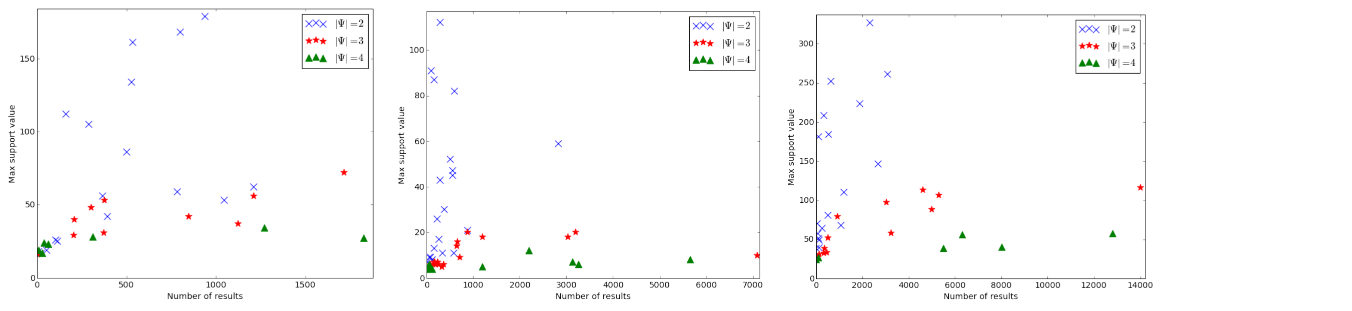
As already explained (see Sections 7.1 and 7.2), there exist various approaches that discover different associations between locations and a given set of keywords. Hence, the purpose of our next experiment was to investigate whether the location sets returned by our approach (STA) are significantly different from those returned by other works, namely collective spatial keyword queries (CSK) and aggregate popularity (AP). We note that we cannot compare with approaches that discover location patterns (LP) as they ignore textual information.

To that end, we computed the top 10 results for STA, AP, and CSK, with respect to the keyword sets we compiled for the three datasets of London, Berlin, and Paris. Then, we computed the Jaccard similarity of the result sets of CSK and AP to ours. This measures the overlap in the query results, i.e., how many location sets STA and either CSK or AP return in common.

The results of this experiment are presented in Table 7.9. The results are averaged across queries with the same keyword set cardinality. As can be observed, the Jaccard similarity scores are very low in all cases, with values not exceeding 0.3. The highest

Table 7.9 Degree of overlap between the associations discovered by STA and those by existing approaches.

	London		Berlin		Paris	
$\|\Psi\|$	**AP**	**CSK**	**AP**	**CSK**	**AP**	**CSK**
2	0.22	0.24	0.28	0.30	0.20	0.14
3	0.17	0.04	0.09	0.07	0.08	0.03
4	0.14	0.03	0.01	0.04	0.00	0.00



(a) London (b) Berlin (c) Paris

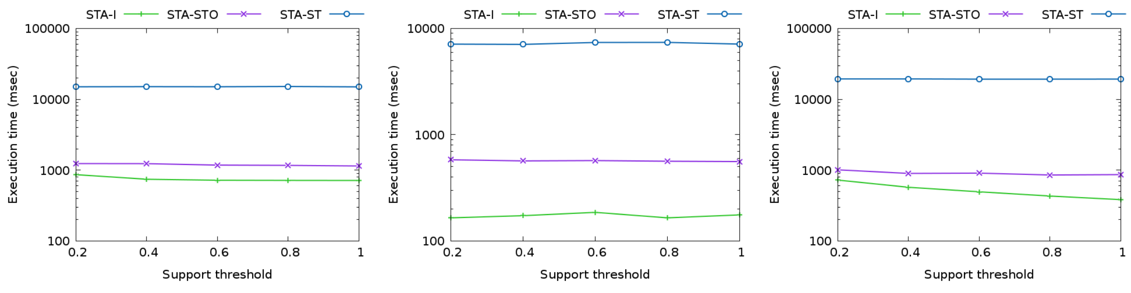
Fig. 7.8 Scatter plots where data points correspond to experiments with distinct keyword sets; the x axis indicates the number of associations above the support threshold and the y axis indicates the highest support among the associations.

scores are observed for queries with 2 keywords, where fewer possible location combinations exist. In those queries, on average, around 2 or 3 of the top 10 location sets discovered by STA are common with those appearing in the results of AP or CSK. The degree of overlap drops even lower when the cardinality of the keyword set increases, allowing for a significantly larger number of candidate location sets. In those cases, often there is only one or zero results in common. This outcome is consistent across the three datasets.

These results show that STA constitutes a novel and distinct criterion for discovering interesting socio-textual associations among locations, which cannot be replicated by existing approaches.

7.7.4 Number of Discovered Associations and Maximum Support

Another aspect to investigate is the distribution of the number of results (associations found) and the support scores for different keyword set cardinalities. To that end, we computed the results for all keyword sets described in Section 7.7.1, i.e., 60 sets



(a) London (b) Berlin (c) Paris

Fig. 7.9 Execution time vs. support threshold; $|\Psi| = 2$.

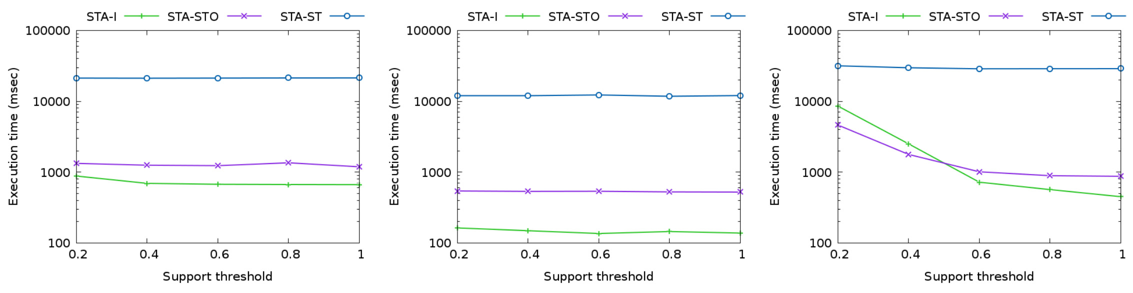
for each dataset, with cardinality $|\Psi| \in [2, 4]$. For each keyword set of the respective dataset, we measured the number of results and the support of the top result. The results of this experiment are shown in Figure 7.8. Note that the value of the support threshold affects both the execution time and the number of results to be found. On the one hand, if the threshold is set too low, an excessive number of results may be returned, and the execution time may also be too high, since only few combinations can be pruned; on the other hand, setting the support threshold too high may return no results.

We notice the following trend in the results for all the three cities. Having only two keywords tends to produce results with high support (e.g., up to around 3% of the total number of users). As the number of keywords increases to 3 or 4, the maximum support among the returned results reduces significantly, dropping close to the support threshold; however, the number of returned results becomes much higher. This is an effect of the fact that, as explained in Section 7.4, the anti-monotonicity property does not hold in our problem.

7.7.5 Evaluation Time

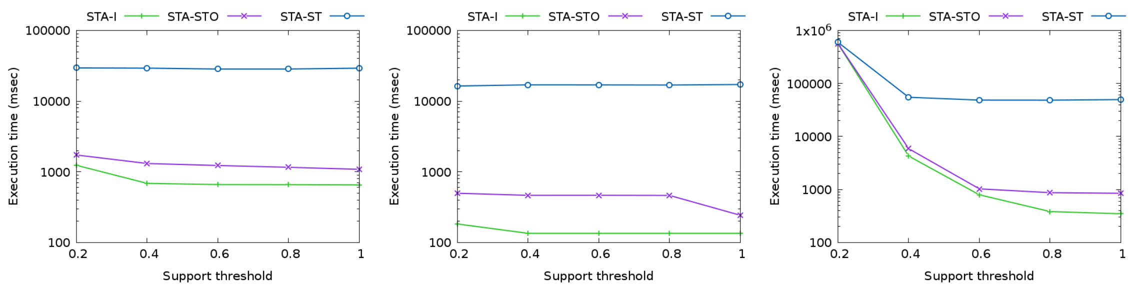
Finally, we evaluate the efficiency of our proposed algorithms. In this experiment, we used the same keyword sets as above.

First, we compare the execution time of the three algorithms, `STA-I`, `STA-ST`, and `STA-STO`, while varying the support threshold parameter σ, which is a percentage of the number of users in each dataset. Note that the basic `STA` method was at least an order of magnitude slower than all other methods and is thus omitted from all plots. Moreover, we include `STA-ST` in the comparison, in order to assess the benefits resulting by the `STA-STO` optimizations. The results are presented in Figures 7.9, 7.10, and 7.11, for 2, 3, and 4 keywords, respectively.



(a) London (b) Berlin (c) Paris

Fig. 7.10 Execution time vs. support threshold; $|\Psi| = 3$.



(a) London (b) Berlin (c) Paris

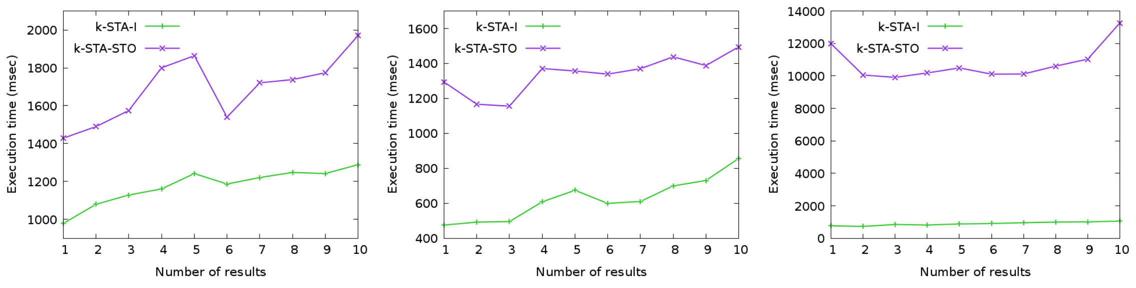
Fig. 7.11 Execution time vs. support threshold; $|\Psi| = 4$.

As the support threshold increases, the performance of all methods improves because fewer location sets survive the pruning. This is apparent in Paris, but not so much in London and Berlin for the specific range of support values depicted. Clearly, `STA-I` achieves the best performance. This is not surprising, since exploiting the preconstructed inverted index saves a substantial amount of the execution time during evaluation. It is worth noticing however that `STA-STO` is also very efficient, achieving competitive execution times compared to `STA-I`. In fact, this is not a merit of the spatio-textual index per se, but rather a result of the proposed optimizations; indeed, the execution times of the generic `STA-ST` are higher by an order of magnitude. The results appear to be consistent across the different datasets and for different number of keywords.

Table 7.10 quantifies the number of location sets (or associations) discovered that have weak support above but actual support below the threshold, which was set to $\sigma = 0.2\%$. For example, in London for $\Psi = 2$, we have that 13.29% of the location sets considered are actual results. As the keyword cardinality increases, the ratio decreases dramatically, because it becomes harder for location sets with weak support above the threshold to also cover all keywords.

Table 7.10 Ratio of number of location sets with support above σ over number of location sets with weak support above σ; $\sigma = 0.2\%$.

$\|\Psi\|$	**London**	**Berlin**	**Paris**
2	13.29%	23.80%	25.98%
3	1.35%	1.09%	3.85%
4	0.01%	0.00%	0.36%



(a) London (b) Berlin (c) Paris

Fig. 7.12 Execution time vs. number of results; $|\Psi| = 3$.

Finally, we evaluate the performance of the algorithms for the top-k version of the problem. The results are presented in Figure 7.12 for $|\Psi| = 3$. A similar outcome is observed, with `K-STA-I` outperforming `K-STA-STO` in all cases. For both algorithms, the execution time tends to increase with k as more results are requested.

7.8 Summary

In this chapter, we have addressed the problem of finding socially and textually associated location sets from user trails derived from geotagged posts. We have formally defined the problem and studied its characteristics. Based on this, we have proposed a general approach for addressing the problem, which we have elaborated to derive three algorithms based on different indexes. Furthermore, we have extended our approach to address also the top-k variant of the problem. The proposed methods have been evaluated experimentally using geotagged Flickr photos in three different cities.

Chapter 8

Summary and Conclusion

Whether it is searching for a restaurant for dinner or finding sights to visit in a new city, location-based search has assumed an important role in our daily lives. Spatial keyword queries provide this ability to search for local information, such as information about POIs, events, news, messages, and photos, by combining spatial and textual predicates into one query. The spatial part of the query is generally a point or a region, whereas the textual part is a set of keywords. Several different types of spatial keyword queries have already been studied in the literature, e.g., the standard queries, collective spatial keyword queries, retrieval of areas of interest, spatio-textual join, etc. Nevertheless, a common feature of these approaches is that their main focus lies in static retrieval, typically of geotagged POI descriptions from different sources, such as Wikipedia, OpenStreetMap, online business directories (e.g., Google My Business), and location-based social networks (e.g., Foursquare).

Recently, online social networks, such as Twitter and Flickr, have emerged as a major source of user-generated spatio-textual data. Instead of the usual static information contained in place descriptions, social networks offer dynamic content in the form of geotagged posts, such as geotagged tweets, geotagged photos, and check-ins, that additionally contains temporal information and is evolving with time. Moreover, due to the massive volume of posts being produced constantly at a rapid pace, challenges and opportunities for efficiently processing and exploring this data arise. Being user-generated, geotagged posts are also a valuable source of information about people's knowledge and opinions regarding places. These characteristics of geotagged posts offer a unique potential for analysis and retrieval.

Thus, in this thesis, we investigated novel techniques for querying and analyzing geotagged posts in social networks. Chapter 1 laid the foundations of our work by outlining our motivation and our goal. Subsequently, we conducted an in-depth survey of related work by suggesting a list of aspects for grouping and reviewing the vast amount of published research on spatial keyword query processing, and discussing how the challenges and techniques studied in this thesis fill the gaps in existing work. After this, Chapters 3–7 presented the problems studied in this thesis and our solutions. For each of these parts, we started by discussing the problem along with any additional background needed, and then presented our approaches and implementation. Finally, through usage examples and experimental evaluation, we demonstrated the merits and limitations of the proposed approaches.

8.1 Summary of Contributions

The contributions of this thesis are fivefold. Driven by the availability of temporal information in posts that is ignored by existing approaches, our first problem (Chapter 3) studied the extension of existing spatio-textual and spatio-temporal indexes to support spatial-temporal-textual filtering of trajectories. However, given the large quantity of available geotagged posts, this plain boolean range filtering can produce a very high number of results that may overwhelm the user. Hence, instead of returning all the matching results, our next approach in Chapter 4 focused on retrieving a selected set of k representative posts for a given spatio-temporal range and keyword filter. Nonetheless, a limitation of this approach is that the results can become quickly obsolete with time as fresh messages are posted. Therefore, in the following part (Chapter 5), we examined the task of continuously maintaining a set of k results for summarizing a stream of posts. Finally, in the last two chapters, we took advantage of the crowdsourced nature of posts for enriching locations with local knowledge and for inferring patterns. First, Chapter 6 presented a system for the discovery and exploration of locally trending topics, i.e., hotspots of keywords occurring frequently in an area. Subsequently, in Chapter 7, we leveraged posts made by mobile users to find sets of places that are thematically associated based on user movement and behavior. Each of these contributions is discussed below in more detail.

8.1.1 Spatial-Temporal-Textual Filtering of Trajectories

An important limitation of existing research in spatial keyword queries is that they ignore temporal information and assume that the objects are static. On the other hand, there has been extensive research in the spatio-temporal database community on retrieving trajectories of moving objects. Spatio-temporal retrieval however typically overlooks any textual information that may be associated with each location update of the moving object. Nevertheless, this information is valuable for various applications that deal with analyzing tracking data of vehicles, ships, and airplanes, where each GPS point might additionally carry some textual information about the current status and destination of the object. This is also true for digital trails generated via user activity on social networks, where each point represents a geotagged post. As a result, in Chapter 3, we focused on the problem of retrieving trajectories of moving objects that are associated with keywords potentially changing with each location update using a spatial-temporal-keyword filter. In particular, we extended two state-of-the-art indexes, one a hybrid index for spatial keyword queries for dealing with temporal information and the other a hybrid index for spatio-temporal queries on trajectories for handling textual data, and compared their performance for the task at hand. Our evaluation of the two methods using two diverse types of datasets, namely yacht movement tracking data and geotagged photos from Flickr, showed that the latter performs better in the majority of our experiments, whereas the former demonstrates a more stable performance, requires less disk space, and performs better on smaller datasets.

8.1.2 Spatial-Temporal-Textual Retrieval of Posts

The focus of the next part of our work also lay on integrating support for temporal information into spatial keyword queries. This is motivated by the observation that the temporal aspect is essential for a variety of applications, including analyzing opinions, topics, and events, and monitoring their evolution over time. Thus, Chapter 4 presented the kCD-STK query for finding a set of top-k results for a given spatio-temporal region and set of keywords. This is achieved by first identifying the objects that lie within the spatio-temporal region and contain the query keywords, and then ranking them based on the combination of two criteria: *spatio-temporal coverage* and *spatio-temporal diversity*. The former promotes results that come from dense regions, whereas the latter ensures that the results are not confined to dense regions only, but are spread over the entire query region. Thus, the fusion of these two measures returns a representative and diverse set of results based on the spatio-temporal distribution of the data. This in turn makes this query suitable for exploratory analysis, particularly for topics and events spanning a large region in space and time. For evaluating the kCD-STK query, we started by deriving a baseline approach, which, however, can be prohibitively expensive as it goes over each post. Thus, we then proposed to extend existing spatio-textual indexes to support temporal information and to exploit these to develop a more efficient index-aware technique for query processing. An experimental evaluation using large real-world datasets of geotagged tweets and photos established the efficiency and superior performance of our proposed optimized method against the baseline algorithm.

8.1.3 Continuous Summarization of Streams of Posts

The problems studied in Chapters 3 and 4 assumed that the results were computed only once in an ad hoc manner and remained valid forever. This assumption is however not always an optimal one for several applications dealing with social network data, where new posts are being generated continuously at a high pace, e.g., monitoring spatial distribution of public opinions and sentiments over time. Thus, in Chapter 5, we studied the problem of continuous spatio-textual summarization of streams of posts. Summarization is an important task in information retrieval and publish/subscribe systems as it provides a quick and succinct overview of a large amount of information through a relatively few documents, which also serve as starting points for exploring the data further. Diversification is a common technique used for generating short yet non-redundant summaries in a large corpus of documents and has been studied extensively in the past. However, since generating an optimal diversified set is an NP-hard task and since the majority of the existing works focus on static collections of documents, diversification in a streaming setting is still an open problem. Therefore, in our approach, we first defined the concepts of *spatio-textual coverage* and *spatio-textual diversity* for generating representative summaries of streams of posts. To ensure that the summaries are current and up-to-date, we used the sliding window model to limit posts to the most recent ones, and devised techniques for updating the summaries dynamically as the window slides. We proposed and evaluated different strategies, with the goal of maximizing the quality of the summary, while minimizing the computation time. To further speed up the computation, we used lightweight structures to group posts based on their spatial and textual attributes. By devising upper bounds on the scores of posts within the groups, we were able to inspect them in a best-first manner and prune non-promising posts early. Through an experimental evaluation of our proposed methods using real-world datasets from Twitter and Flickr, we demonstrated that our methods and optimizations can be used efficiently to continuously maintain concise summaries of streams of posts.

8.1.4 Discovery and Exploration of Locally Trending Topics

Geotagged posts are a valuable source of information about people's knowledge and opinions about places. Moreover, given the huge volume of available content and the inherent noise in crowdsourced data, identifying potentially interesting information posted daily on social networks is an important as well as challenging task. Consequently, there has been a significant amount of work on finding popular topics among posts and presenting these to users. Location is an important aspect here since topics, opinions, and events tend to vary across different regions. Thus, in this part of our work, we developed a prototype to extract locally trending topics worldwide continuously over a stream of posts using a sliding window. Moreover, to allow users to quickly grasp the context or background of a topic, the system allows users to retrieve a small set of representative messages related to it. In addition to visualizing this information on spatial, textual, and temporal dimensions, the application provides other mechanisms to explore and dig deeper into the dataset, such as spatial-temporal-textual similarity-based retrieval and filtering, and iterative drill down. Chapter 6 presented the architecture of the system and described each of its main components in detail, including the *Storage System*, the *Topic Detection* module, the *Topic Summarization* module, the *Post Similarity* module, and the *web-based user interface*. Finally, the functionality of system was demonstrated using a continuously updated dataset of more than 80 million geotagged Twitter messages and by going through a typical usage scenario.

8.1.5 Mining Associated Location Sets

Our motivation for this part of our work was along the lines of that of the previous one (Chapter 6), namely that a post at a certain location generally says something about that location. Thus, collectively, a corpus of geotagged posts adds a dimension of crowdsourced intelligence to places that can be examined to reveal insights and patterns regarding people's knowledge and interactions with places. In particular, there has been a significant amount of research in the area of mining mobility patterns to discover sets of locations appearing together frequently in user trails. Here, the objective is to utilize user-generated content to understand how people move in a region and consequently, to provide them with better local infrastructure and services. Similarly, in spatial keyword query processing, collective spatial keyword queries find groups of objects that together satisfy user requirements specified by a given set of keywords and that are close to each other. The motivation here is that user requirements can often be complex, and thus a single location might not suffice. In mobility pattern mining, the locations are only socially associated as textual information is ignored, whereas in collective spatial keyword queries, the locations are textually, but not socially associated. Thus, the work in Chapter 7 filled the gaps in these two areas of research by investigating the problem of finding location sets for a given set of keywords that are associated both socially and textually. The social condition makes certain that the locations co-occur in user trails derived from social networks, whereas the textual criterion ensures that the posts made by users at the locations are collectively relevant to the query keywords. This allows us to leverage users' mobility patterns and their semantic characterization of locations as evidence to identify places that are thematically associated. In our analysis, we started out by formally defining the problem and studying its characteristics. This led us to the observation that although our problem appears similar to the task of mining frequent itemsets, one cannot utilize an Apriori-like algorithm directly in our scenario. Thus, we proposed the concepts of *weak support* and *relevant user* that allowed us to use a frequent itemset mining algorithm for our problem. Based on this, we proposed three different approaches: a *baseline approach* without an underlying index, an *index-aware approach* that operates over a simple inverted index, and a *spatio-textual index-based approach*. Moreover, we proposed algorithms for both the threshold-based and the top-*k* variants of the problem. Our experimental evaluation using geotagged photos from three different cities showed that our index-aware methods outperform the baseline approach by a large margin.

8.2 Outlook

There are several directions in which the work presented in this thesis can be extended. Below we outline the ones that we consider as most promising and challenging.

8.2.1 Distributed Processing

Most of the existing works on spatial keyword queries use sequential processing of data and assume that it can fit in the main memory of a single machine during index construction and query processing. This is also reflected in the sizes of datasets used for experimental evaluation, making it uncertain how these techniques would cope with larger amounts of data or with data partitioned across multiple machines. On the other hand, due to the availability of spatio-textual data at an unprecedented scale, research into distributed and parallel processing methods is gaining momentum [116]. Distributed computing frameworks, such as Hadoop and Spark based on the MapReduce programming model, offer potential solutions for handling this data deluge. However, developing systems [117, 177, 80] and extending existing techniques [9, 115, 136, 49, 123] for handling spatio-textual and spatio-temporal data based on these paradigms are still open challenges that have received limited attention so far. Thus, there is a significant scope for future work in this area, including the adaptation and extension of problems presented in this thesis to these settings.

8.2.2 Standardized Benchmarks and Surveys

The earliest survey of spatial keyword query processing methods evaluated and compared twelve spatio-textual indexes for standard queries [32]. Since then, as is evident from our survey of existing research in Chapter 2, research in this area has expanded far beyond the standard queries and several different query formulations and query processing methods have been proposed. Moreover, each of these methods evaluates its contributions in a different setting against different chosen baseline methods. As a result, due to the overwhelming number of published works and due to their evaluation in separate environments, it is becoming increasingly difficult to identify the merits and shortcomings of a specific technique, and to compare them against others. Thus, standardized benchmarks and surveys for methodically evaluating and comparing these techniques is a very important direction of future research that has received very little attention so far.

8.2.3 Integration into Mainstream Databases and GIS Tools

The bulk of the different query types and evaluation techniques proposed in existing literature use tailor-made hybrid indexes and data structures for improving query processing times. Since these methods are custom-built and target specific problems only, it is uncertain how they might perform in other contexts, and consequently their adoption in mainstream databases and GIS tools has been limited so far. Although open-source text retrieval systems, such as Lucene[1] and ElasticSearch[2], now come with integrated support for spatial search, the range of spatio-textual queries supported by them is very limited. Furthermore, with the exception of these sys-

[1]https://lucene.apache.org/
[2]http://www.elastic.co/products/elasticsearch

tems, open-source implementations of spatial keyword query processing methods are scarce. Thus, an important direction of future work lies in developing a unified general-purpose framework comprising multiple access methods and techniques, such as those presented in this thesis, and integrating it into existing open-source databases and GIS tools.

REFERENCES

[1] H. Abdelhaq, C. Sengstock, and M. Gertz. Eventweet: Online localized event detection from Twitter. In *Proceedings of the VLDB Endowment*, pages 1326–1329. VLDB Endowment, 2013. (Cited on page 100.)

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of 20th International Conference on Very Large Data Bases*, pages 487–499. VLDB Endowment, 1994. (Cited on pages 9 and 115.)

[3] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009. (Cited on pages 56 and 57.)

[4] R. Ahuja, N. Armenatzoglou, D. Papadias, and G. J. Fakas. Geo-social keyword search. In *Proceedings of the 14th International Symposium on Advances in Spatial and Temporal Databases*, pages 431–450. Springer, 2015. (Cited on pages 18 and 27.)

[5] O. Alonso and K. Shiells. Timelines as summaries of popular scheduled events. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1037–1044. ACM, 2013. (Cited on page 58.)

[6] O. Alonso, J. Strötgen, R. A. Baeza-Yates, and M. Gertz. Temporal information retrieval: Challenges and opportunities. In *Proceedings of the 1st International Temporal Web Analytics Workshop*, pages 1–8, 2011. (Cited on page 58.)

[7] A. Anand, S. Bedathur, K. Berberich, and R. Schenkel. Index maintenance for time-travel text search. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 235–244. ACM, 2012. (Cited on page 58.)

[8] I. Arikan, S. J. Bedathur, and K. Berberich. Time will tell: Leveraging temporal expressions in IR. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*. ACM, 2009. (Cited on page 58.)

[9] J. Ballesteros, A. Cary, and N. Rishe. SpSJoin: Parallel spatial similarity joins. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 481–484. ACM, 2011. (Cited on pages 18, 25, 29, and 151.)

[10] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th International Conference on World Wide Web*, pages 131–140. ACM, 2007. (Cited on page 30.)

[11] M. Becker, P. Singer, F. Lemmerich, A. Hotho, D. Helic, and M. Strohmaier. Photowalking the city: Comparing hypotheses about urban photo trails on Flickr. In *Proceedings of the 7th International Conference on Social Informatics*, pages 227–244, 2015. (Cited on page 117.)

[12] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *ACM SIGMOD Record*, number 2, pages 322–331. ACM, 1990. (Cited on page 19.)

[13] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. (Cited on page 14.)

[14] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 519–526. ACM, 2007. (Cited on page 58.)

[15] B. E. Birnbaum and K. J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 49–60. Springer, 2006. (Cited on pages 63 and 84.)

[16] P. Bouros, S. Ge, and N. Mamoulis. Spatio-textual similarity joins. In *Proceedings of the VLDB Endowment*, pages 1–12. VLDB Endowment, 2012. (Cited on pages 18, 29, and 30.)

[17] C. Budak, T. Georgiou, D. Agrawal, and A. El Abbadi. GeoScope: Online detection of geo-correlated information trends in social networks. In *Proceedings of the VLDB Endowment*, pages 229–240. VLDB Endowment, 2013. (Cited on page 100.)

[18] G. Cai, C. Hio, L. Bermingham, K. Lee, and I. Lee. Mining frequent trajectory patterns and regions-of-interest from Flickr photos. In *47th Hawaii International Conference on System Sciences*, pages 1454–1463. IEEE, 2014. (Cited on pages 112, 113, and 117.)

[19] R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. Survey of temporal information retrieval and related applications. *ACM Computing Surveys*, 47(2):15, 2015. (Cited on page 58.)

[20] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. In *Proceedings of the VLDB Endowment*, pages 373–384. VLDB Endowment, 2010. (Cited on pages 4, 18, and 21.)

[21] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 373–384. ACM, 2011. (Cited on pages 3, 4, 18, 19, and 113.)

[22] X. Cao, L. Chen, G. Cong, C. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. Yiu. Spatial keyword querying. *Conceptual Modeling*, pages 16–29, 2012. (Cited on pages 3 and 11.)

[23] X. Cao, G. Cong, C. S. Jensen, and M. L. Yiu. Retrieving regions of interest for user exploration. In *Proceedings of the VLDB Endowment*, pages 733–744. VLDB Endowment, 2014. (Cited on pages 4, 18, 19, and 28.)

[24] X. Cao, G. Cong, T. Guo, C. S. Jensen, and B. C. Ooi. Efficient processing of spatial group keyword queries. *ACM Transactions on Database Systems*, 40(2): 13, 2015. (Cited on pages 3, 18, and 19.)

[25] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336. ACM, 1998. (Cited on pages 56, 57, and 80.)

[26] A. Cary, O. Wolfson, and N. Rishe. Efficient and scalable method for processing top-k spatial boolean queries. In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management*, pages 87–95. Springer, 2010. (Cited on pages 12 and 25.)

[27] M. Ceccarello, A. Pietracaprina, G. Pucci, and E. Upfal. MapReduce and streaming algorithms for diversity maximization in metric spaces of bounded doubling dimension. In *Proceedings of the VLDB Endowment*, pages 469–480. VLDB Endowment, 2017. (Cited on page 80.)

[28] V. P. Chakka, A. Everspaugh, and J. M. Patel. Indexing large trajectory data sets with SETI. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research*, 2003. (Cited on pages 35 and 37.)

[29] D. Chakrabarti and K. Punera. Event summarization using tweets. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*, pages 66–73. AAAI Press, 2011. (Cited on page 100.)

[30] L. Chen and G. Cong. Diversity-aware top-k publish/subscribe for text stream. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 347–362. ACM, 2015. (Cited on pages 23 and 99.)

[31] L. Chen, G. Cong, and X. Cao. An efficient query indexing mechanism for filtering geo-textual data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 749–760. ACM, 2013. (Cited on pages 18, 22, 23, and 99.)

[32] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: An experimental evaluation. In *Proceedings of the VLDB Endowment*, pages 217–228. VLDB Endowment, 2013. (Cited on pages xv, 3, 11, 12, 13, 15, 16, and 151.)

[33] L. Chen, Y. Cui, G. Cong, and X. Cao. SOPS: A system for efficient processing of spatial-keyword publish/subscribe. In *Proceedings of the VLDB Endowment*, pages 1601–1604. VLDB Endowment, 2014. (Cited on pages 18 and 23.)

[34] L. Chen, G. Cong, X. Cao, and K.-L. Tan. Temporal spatial-keyword top-k publish/subscribe. In *Proceedings of the IEEE 31st International Conference on Data Engineering*, pages 255–266. IEEE, 2015. (Cited on pages 18, 23, 58, 68, and 93.)

[35] Y. Chen, H. Amiri, Z. Li, and T.-S. Chua. Emerging topic detection for organizations from microblogs. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2013. (Cited on page 6.)

[36] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 277–288. ACM, 2006. (Cited on pages 12, 18, and 27.)

[37] Z. Chen, G. Cong, Z. Zhang, T. Z. Fuz, and L. Chen. Distributed publish/subscribe query processing on the spatio-textual data stream. In *Proceedings of the IEEE 33rd International Conference on Data Engineering*, pages 1095–1106. IEEE, 2017. (Cited on pages 18 and 25.)

[38] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel. Text vs. space: Efficient geo-search query processing. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 423–432. ACM, 2011. (Cited on pages 12, 14, 15, 37, and 41.)

[39] G. Cong and C. S. Jensen. Querying geo-textual data: Spatial keyword queries and beyond. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data*, pages 2207–2212. ACM, 2016. (Cited on pages 3 and 11.)

[40] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. In *Proceedings of the VLDB Endowment*, pages 337–348. VLDB Endowment, 2009. (Cited on pages 12 and 15.)

[41] G. Cong, H. Lu, B. C. Ooi, D. Zhang, and M. Zhang. Efficient spatial keyword search in trajectory databases. *arXiv preprint arXiv:1205.2880*, 2012. (Cited on pages 18, 26, and 34.)

[42] G. Cong, K. Feng, and K. Zhao. Querying and mining geo-textual data for exploration: Challenges and opportunities. In *Proceedings of the IEEE 32nd International Conference on Data Engineering Workshops*, pages 165–168. IEEE, 2016. (Cited on pages 3 and 11.)

[43] D. J. Crandall, L. Backstrom, D. P. Huttenlocher, and J. M. Kleinberg. Mapping the world's photos. In *Proceedings of the 18th International Conference on World Wide Web*, pages 761–770. ACM, 2009. (Cited on page 117.)

[44] V. T. de Almeida and R. H. Güting. Indexing the trajectories of moving objects in networks. *GeoInformatica*, 9(1):33–60, 2005. (Cited on page 36.)

[45] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, pages 35–44. ACM, 2010. (Cited on page 117.)

[46] I. De Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, pages 656–665. IEEE, 2008. (Cited on page 12.)

[47] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. (Cited on page 20.)

[48] Z. Dou, S. Hu, K. Chen, R. Song, and J.-R. Wen. Multi-dimensional search result diversification. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 475–484. ACM, 2011. (Cited on pages 56 and 57.)

[49] C. Doulkeridis, A. Vlachou, D. Mpestas, and N. Mamoulis. Parallel and distributed processing of spatial preference queries using keywords. In *Proceedings of the 20th International Conference on Extending Database Technology*, pages 318–329, 2017. (Cited on pages 18, 20, 25, and 151.)

[50] M. Drosou and E. Pitoura. Search result diversification. *ACM SIGMOD Record*, 39(1):41–47, 2010. (Cited on pages 56, 57, 78, and 80.)

[51] M. Drosou and E. Pitoura. Diverse set selection over dynamic data. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1102–1116, 2014. (Cited on pages 78, 81, and 85.)

[52] M. Drosou and E. Pitoura. Multiple radii disc diversity: Result diversification based on dissimilarity and coverage. *ACM Transactions on Database Systems*, 40(1):4, 2015. (Cited on pages 55, 56, 57, and 81.)

[53] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the Web. In *Proceedings of the 10th International Conference on World Wide Web*, pages 613–622. ACM, 2001. (Cited on page 112.)

[54] C. Efstathiades, A. Belesiotis, D. Skoutas, and D. Pfoser. Similarity search on spatio-textual point sets. In *Proceedings of the 19th International Conference on Extending Database Technology*, pages 329–340, 2016. (Cited on pages 18 and 30.)

[55] A. Eldawy and M. F. Mokbel. A demonstration of SpatialHadoop: An efficient MapReduce framework for spatial data. pages 1230–1233. VLDB Endowment, 2013. (Cited on page 24.)

[56] A. Eldawy and M. F. Mokbel. SpatialHadoop: A MapReduce framework for spatial data. In *Proceedings of the IEEE 31st International Conference on Data Engineering*, pages 1352–1363. IEEE, 2015. (Cited on page 24.)

[57] A. Eldawy, L. Alarabi, and M. F. Mokbel. Spatial partitioning techniques in SpatialHadoop. In *Proceedings of the VLDB Endowment*, pages 1602–1605. VLDB Endowment, 2015. (Cited on page 24.)

[58] A. Eldawy, M. F. Mokbel, et al. The era of big spatial data: A survey. *Foundations and Trends® in Databases*, 6(3-4):163–273, 2016. (Cited on page 24.)

[59] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656, 2003. (Cited on page 17.)

[60] J. Fan, G. Li, L. Zhou, S. Chen, and J. Hu. SEAL: Spatio-textual similarity search. In *Proceedings of the VLDB Endowment*, pages 824–835. VLDB Endowment, 2012. (Cited on pages 18, 27, and 28.)

[61] K. Feng, G. Cong, S. S. Bhowmick, W.-C. Peng, and C. Miao. Towards best region search for data exploration. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data*, pages 1055–1070. ACM, 2016. (Cited on pages 4, 18, and 19.)

[62] K. Feng, K. Zhao, Y. Liu, and G. Cong. A system for region search and exploration. In *Proceedings of the VLDB Endowment*, pages 1549–1552. VLDB Endowment, 2016. (Cited on pages 4, 18, and 19.)

[63] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang. STREAMCUBE: Hierarchical spatio-temporal hashtag clustering for event exploration over the Twitter stream. In *Proceedings of the IEEE 31st International Conference on Data Engineering*, pages 1561–1572. IEEE, 2015. (Cited on page 100.)

[64] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974. (Cited on page 14.)

[65] A. Fox, C. Eichelberger, J. Hughes, and S. Lyon. Spatio-temporal indexing in non-relational distributed databases. In *Proceedings of the 2013 IEEE International Conference on Big Data*, pages 291–299. IEEE, 2013. (Cited on page 24.)

[66] E. Frentzos. Indexing objects moving on fixed networks. In *Proceedings of the 8th International Symposium on Advances in Spatial and Temporal Databases*, pages 289–305. Springer, 2003. (Cited on page 36.)

[67] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Algorithms for nearest neighbor search on moving object trajectories. *Geoinformatica*, 11(2):159–193, 2007. (Cited on page 33.)

[68] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998. (Cited on page 14.)

[69] Y. Gao, X. Qin, B. Zheng, and G. Chen. Efficient reverse top-k boolean spatial keyword queries on road networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1205–1218, 2015. (Cited on pages 18, 28, and 29.)

[70] Y. Gao, J. Zhao, B. Zheng, and G. Chen. Efficient collective spatial keyword query processing on road networks. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):469–480, 2016. (Cited on pages 3, 18, 19, and 28.)

[71] Y.-J. Gao, C. Li, G.-C. Chen, L. Chen, X.-T. Jiang, and C. Chen. Efficient k-nearest-neighbor search algorithms for historical moving object trajectories. *Journal of Computer Science and Technology*, 22(2):232–244, 2007. (Cited on page 33.)

[72] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz. Multi-document summarization by sentence extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization*, pages 40–48. Association for Computational Linguistics, 2000. (Cited on page 82.)

[73] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th International Conference on World Wide Web*, pages 381–390. ACM, 2009. (Cited on pages 28, 55, 56, 57, 78, 79, and 80.)

[74] S. Grin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998. (Cited on page 21.)

[75] L. Guo, J. Shao, H. H. Aung, and K.-L. Tan. Efficient continuous top-k spatial keyword queries on road networks. *GeoInformatica*, 19(1):29–60, 2015. (Cited on pages 18, 22, and 28.)

[76] L. Guo, D. Zhang, G. Li, K.-L. Tan, and Z. Bao. Location-aware pub/sub system: When continuous moving queries meet dynamic event streams. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 843–857. ACM, 2015. (Cited on pages 18 and 22.)

[77] T. Guo, X. Cao, and G. Cong. Efficient algorithms for answering the m-closest keywords query. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 405–418. ACM, 2015. (Cited on pages 3, 18, and 19.)

[78] R. H. Güting, T. Behr, and J. Xu. Efficient k-nearest neighbor search on moving object trajectories. *The VLDB Journal*, 19(5):687–714, 2010. (Cited on page 33.)

[79] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57. ACM, 1984. (Cited on page 14.)

[80] S. Hagedorn and T. Räth. Efficient spatio-temporal event processing with STARK. In *Proceedings of the 20th International Conference on Extending Database Technology*, pages 570–573, 2017. (Cited on page 151.)

[81] S. Hagedorn, P. Götze, and K.-U. Sattler. Big spatial data processing frameworks: Feature and performance evaluation. In *Proceedings of the 20th International Conference on Extending Database Technology*, pages 490–493, 2017. (Cited on page 24.)

[82] Y. Han, L. Wang, Y. Zhang, W. Zhang, and X. Lin. Spatial keyword range search on trajectories. In *Proceedings of the 20th International Conference on Database Systems for Advanced Applications*, pages 223–240. Springer, 2015. (Cited on page 34.)

[83] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management*, pages 16–16. IEEE, 2007. (Cited on pages 12, 37, and 38.)

[84] A. M. Hendawi and M. F. Mokbel. Predictive spatio-temporal queries: A comprehensive survey and future directions. In *Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, pages 97–104. ACM, 2012. (Cited on page 33.)

[85] D. Hilbert. Ueber die stetige Abbildung einer line auf ein Flächenstück. *Mathematische Annalen*, 38(3):459–460, 1891. (Cited on page 14.)

[86] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 24(2):265–318, 1999. (Cited on pages 15 and 132.)

[87] T.-A. Hoang-Vu, H. T. Vo, and J. Freire. A unified index for spatio-temporal keyword queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 135–144. ACM, 2016. (Cited on pages 18 and 26.)

[88] H. Hu, Y. Liu, G. Li, J. Feng, and K.-L. Tan. A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions. In *Proceedings of the IEEE 31st International Conference on Data Engineering*, pages 711–722. IEEE, 2015. (Cited on pages 18 and 23.)

[89] W. Huang, G. Li, K.-L. Tan, and J. Feng. Efficient safe-region construction for moving top-k spatial keyword queries. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 932–941. ACM, 2012. (Cited on pages 18 and 22.)

[90] P. Indyk, S. Mahabadi, M. Mahdian, and V. S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 100–108. ACM, 2014. (Cited on page 80.)

[91] G. S. Iwerks, H. Samet, and K. Smith. Continuous k-nearest neighbor queries for continuously moving points with updates. In *Proceedings of the 29th International Conference on Very Large Data Bases*, pages 512–523. VLDB Endowment, 2003. (Cited on page 33.)

[92] A. Jatowt, É. Antoine, Y. Kawai, and T. Akiyama. Mapping temporal horizons: Analysis of collective future and past related attention in Twitter. In *Proceedings of the 24th International Conference on World Wide Web*, pages 484–494. ACM, 2015. (Cited on page 58.)

[93] C. S. Jensen, H. Lu, and B. Yang. Indexing the trajectories of moving objects in symbolic indoor space. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*, pages 208–227. Springer, 2009. (Cited on page 36.)

[94] J. Jiang, H. Lu, B. Yang, and B. Cui. Finding top-k local users in geo-tagged social media data. In *Proceedings of the IEEE 31st International Conference on Data Engineering*, pages 267–278. IEEE, 2015. (Cited on pages 18 and 26.)

[95] P. Jin, J. Lian, X. Zhao, and S. Wan. TISE: A temporal search engine for web contents. In *Proceedings of the Second International Symposium on Intelligent Information Technology Application*, pages 220–224. IEEE, 2008. (Cited on page 58.)

[96] N. Kanhabua and W. Nejdl. Understanding the diversity of tweets in the time of outbreaks. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1335–1342. ACM, 2013. (Cited on page 6.)

[97] A. Khodaei, C. Shahabi, and C. Li. Hybrid indexing and seamless ranking of spatial and textual features of web documents. In *Proceedings of the 21st International Conference on Database and Expert Systems Applications*, pages 450–466. Springer, 2010. (Cited on page 12.)

[98] S. Kisilevich, D. A. Keim, and L. Rokach. A novel approach to mining travel sequences using collections of geotagged photos. In *Geospatial Thinking - International AGILE'2010 Conference*, pages 163–182. Springer, 2010. (Cited on pages 112, 113, and 116.)

[99] F. Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 201–212. ACM, 2000. (Cited on page 29.)

[100] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*, pages 579–588. ACM, 2010. (Cited on page 117.)

[101] T. T. T. Le and B. G. Nickerson. Efficient search of moving objects on a planar graph. In *Proceedings of the 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, page 41. ACM, 2008. (Cited on page 36.)

[102] I. Lee, G. Cai, and K. Lee. Mining points-of-interest association rules from geo-tagged photos. In *46th Hawaii International Conference on System Sciences*, pages 1580–1588. IEEE, 2013. (Cited on pages 112, 113, and 116.)

[103] G. Li, J. Feng, and J. Xu. DESKS: Direction-aware spatial keyword search. In *Proceedings of the IEEE 28th International Conference on Data Engineering*, pages 474–485. IEEE, 2012. (Cited on pages 18 and 30.)

[104] G. Li, Y. Wang, T. Wang, and J. Feng. Location-aware publish/subscribe. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 802–810. ACM, 2013. (Cited on pages 18 and 23.)

[105] J. Li, D. Maier, K. Tufte, V. Papadimos, and P. A. Tucker. Semantics and evaluation techniques for window aggregates in data streams. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 311–322. ACM, 2005. (Cited on pages 82 and 85.)

[106] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving object clusters. In *Proceedings of the VLDB Endowment*, pages 723–734. VLDB Endowment, 2010. (Cited on page 33.)

[107] Z. Li, K. C. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang. IR-tree: An efficient index for geographic document search. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):585–599, 2011. (Cited on page 12.)

[108] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520. Association for Computational Linguistics, 2011. (Cited on pages 80 and 82.)

[109] S. Liu, G. Li, and J. Feng. Star-Join: Spatio-textual similarity join. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2194–2198. ACM, 2012. (Cited on pages 18, 27, and 30.)

[110] S. Liu, G. Li, and J. Feng. A prefix-filter based method for spatio-textual similarity join. *IEEE Transactions on Knowledge and Data Engineering*, 26(10): 2354–2367, 2014. (Cited on pages 18, 27, and 30.)

[111] P. Longley. *Geographic information systems and science*. John Wiley & Sons, 2005. (Cited on page 89.)

[112] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 349–360. ACM, 2011. (Cited on pages 18 and 29.)

[113] X. Lu, C. Wang, J. Yang, Y. Pang, and L. Zhang. Photo2Trip: Generating travel routes from geo-tagged photos for trip planning. In *Proceedings of the 18th International Conference on Multimedia 2010*, pages 143–152, 2010. (Cited on page 117.)

[114] Y. Lu, M. Zhang, S. Witherspoon, Y. Yesha, Y. Yesha, and N. Rishe. SksOpen: Efficient indexing, querying, and visualization of geo-spatial big data. In *Proceedings of the IEEE 12th International Conference on Machine Learning and Applications*, pages 495–500. IEEE, 2013. (Cited on pages 18 and 25.)

[115] S. Luo, Y. Luo, S. Zhou, G. Cong, J. Guan, and Z. Yong. Distributed spatial keyword querying on road networks. In *Proceedings of the 17th International Conference on Extending Database Technology*, pages 235–246, 2014. (Cited on pages 18, 25, 28, and 151.)

[116] A. Mahmood and W. G. Aref. Query processing techniques for big spatial-keyword data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1777–1782. ACM, 2017. (Cited on pages 24 and 151.)

[117] A. R. Mahmood, A. M. Aly, T. Qadah, E. K. Rezig, A. Daghistani, A. Madkour, A. S. Abdelhamid, M. S. Hassan, W. G. Aref, and S. Basalamah. Tornado: A distributed spatio-textual stream processing system. In *Proceedings of the VLDB Endowment*, pages 2020–2023. VLDB Endowment, 2015. (Cited on pages 18, 24, and 151.)

[118] A. Majid, L. Chen, G. Chen, H. T. Mirza, I. Hussain, and J. Woodward. A context-aware personalized travel recommendation system based on geotagged social media data mining. *International Journal of Geographical Information Science*, 27(4):662–684, 2013. (Cited on page 117.)

[119] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge University Press, 2008. (Cited on page 83.)

[120] P. Mehta, D. Skoutas, and A. Voisard. Spatio-temporal keyword queries for moving objects. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 55. ACM, 2015. (Cited on page 6.)

[121] P. Mehta, D. Sacharidis, D. Skoutas, and A. Voisard. Keyword-based retrieval of frequent location sets in geotagged photo trails. In *Proceedings of the 8th ACM Conference on Web Science*, pages 348–349. ACM, 2016. (Cited on page 8.)

[122] P. Mehta, D. Skoutas, D. Sacharidis, and A. Voisard. Coverage and diversity aware top-k query for spatio-temporal posts. In *Proceedings of the 24th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 19. ACM, 2016. (Cited on pages 6, 101, and 105.)

[123] P. Mehta, C. Windolf, and A. Voisard. Spatio-temporal hotspot computation on Apache Spark (GIS Cup). In *Proceedings of the 24th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2016. (Cited on page 151.)

[124] P. Mehta, M. Kotlarski, D. Skoutas, D. Sacharidis, K. Patroumpas, and A. Voisard. μTOP: Spatio-temporal detection and summarization of locally trending topics in microblog posts. In *Proceedings of the 20th International Conference on Extending Database Technology*, pages 558–561, 2017. (Cited on page 8.)

[125] P. Mehta, D. Sacharidis, D. Skoutas, and A. Voisard. Finding socio-textual associations among locations. In *Proceedings of the 20th International Conference on Extending Database Technology*, pages 120–131, 2017. (Cited on page 8.)

[126] E. Minack, W. Siberski, and W. Nejdl. Incremental diversification for very large sets: A streaming-based approach. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 585–594. ACM, 2011. (Cited on pages 78, 81, 85, 86, and 87.)

[127] M. F. Mokbel, T. M. Ghanem, and W. G. Aref. Spatio-temporal access methods. *IEEE Data(base) Engineering Bulletin*, 26(2):40–49, 2003. (Cited on page 35.)

[128] G. M. Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company New York, 1966. (Cited on page 14.)

[129] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 635–646. ACM, 2006. (Cited on page 23.)

[130] S. Nepomnyachiy, B. Gelley, W. Jiang, and T. Minkus. What, where, and when: Keyword search with spatio-temporal ranges. In *Proceedings of the 8th Workshop on Geographic Information Retrieval*, page 2. ACM, 2014. (Cited on pages 18, 26, 34, and 58.)

[131] L. Nguyen-Dinh, W. G. Aref, and M. F. Mokbel. Spatio-temporal access methods: Part 2 (2003 - 2010). *IEEE Data(base) Engineering Bulletin*, 33(2):46–55, 2010. (Cited on pages 33 and 35.)

[132] J. Ni and C. V. Ravishankar. PA-tree: A parametric indexing scheme for spatio-temporal trajectories. In *Proceedings of the 9th International Symposium on Advances in Spatial and Temporal Databases*, pages 254–272. Springer, 2005. (Cited on page 36.)

[133] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 9 (1):38–71, 1984. (Cited on page 14.)

[134] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of the 29th International Conference on Very Large Data Bases*, pages 802–813. VLDB Endowment, 2003. (Cited on page 28.)

[135] K. Patroumpas and M. Loukadakis. Monitoring spatial coverage of trending topics in Twitter. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, page 7. ACM, 2016. (Cited on pages 100 and 103.)

[136] J. Rao, J. Lin, and H. Samet. Partitioning strategies for spatio-textual similarity join. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 40–49. ACM, 2014. (Cited on pages 18, 30, and 151.)

[137] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994. (Cited on pages 63 and 84.)

[138] A. Ritter, O. Etzioni, S. Clark, et al. Open domain event extraction from Twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1104–1112. ACM, 2012. (Cited on page 6.)

[139] J. B. Rocha-Junior and K. Nørvåg. Top-k spatial keyword queries on road networks. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 168–179. ACM, 2012. (Cited on pages 18 and 28.)

[140] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In *Proceedings of the 12th International Symposium on Advances in Spatial and Temporal Databases*, pages 205–222. Springer, 2011. (Cited on page 12.)

[141] D. Sacharidis, P. Mehta, D. Skoutas, K. Patroumpas, and A. Voisard. Continuous summarization of streaming spatio-textual posts. Submitted for publication to the *25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2017. (Cited on page 7.)

[142] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988. (Cited on page 83.)

[143] H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 0123694469. (Cited on page 42.)

[144] B. Sharifi, M.-A. Hutton, and J. Kalita. Summarizing microblogs automatically. In *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 685–688. Association for Computational Linguistics, 2010. (Cited on page 100.)

[145] G. Skoumas, D. Skoutas, and A. Vlachaki. Efficient identification and approximation of k-nearest moving neighbors. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 264–273. ACM, 2013. (Cited on page 33.)

[146] D. Skoutas, D. Sacharidis, and K. Stamatoukos. Identifying and describing streets of interest. In *Proceedings of the 19th International Conference on Extending Database Technology*, pages 437–448, 2016. (Cited on pages 4, 17, 18, and 28.)

[147] E. Spyrou, I. Sofianos, and P. Mylonas. Mining tourist routes from Flickr photos. In *Proceedings of the 10th International Workshop on Semantic and Social Media Adaptation and Personalization*, pages 1–5. IEEE, 2015. (Cited on pages 112, 113, and 116.)

[148] Y. Sun, H. Fan, M. Bakillah, and A. Zipf. Road-based travel recommendation using geo-tagged images. *Computers, Environment and Urban Systems*, 53: 110–122, 2015. (Cited on page 117.)

[149] C. Tai, D. Yang, L. Lin, and M. Chen. Recommending personalized scenic itinerary with geo-tagged photos. In *Proceedings of the 2008 IEEE International Conference on Multimedia and Expo*, pages 1209–1212. IEEE, 2008. (Cited on page 117.)

[150] H. Takamura and M. Okumura. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789. Association for Computational Linguistics, 2009. (Cited on page 80.)

[151] M. Tang, Y. Yu, Q. M. Malluhi, M. Ouzzani, and W. G. Aref. LocationSpark: A distributed in-memory data management system for big spatial data. In *Proceedings of the VLDB Endowment*, pages 1565–1568. VLDB Endowment, 2016. (Cited on page 24.)

[152] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 287–298. VLDB Endowment, 2002. (Cited on page 33.)

[153] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 1(8), 2015. (Cited on pages 46, 69, 93, and 136.)

[154] G. Tsatsanifos and A. Vlachou. On processing top-k spatio-textual preference queries. In *Proceedings of the 18th International Conference on Extending Database Technology*, pages 433–444, 2015. (Cited on pages 4, 18, and 20.)

[155] S. Vaid, C. B. Jones, H. Joho, and M. Sanderson. Spatio-textual indexing for geographical search on the Web. In *Proceedings of the 9th International Symposium on Advances in Spatial and Temporal Databases*, pages 218–235. Springer, 2005. (Cited on page 12.)

[156] G. Valkanas and D. Gunopulos. How the live Web feels about events. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 639–648. ACM, 2013. (Cited on page 6.)

[157] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. A. Yahia. Efficient computation of diverse query results. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, pages 228–236. IEEE, 2008. (Cited on page 57.)

[158] M. R. Vieira, H. L. Razente, M. C. Barioni, M. Hadjieleftheriou, D. Srivastava, C. Traina, and V. J. Tsotras. On query result diversification. In *Proceedings of the IEEE 27th International Conference on Data Engineering*, pages 1163–1174. IEEE, 2011. (Cited on pages 56, 57, 78, and 80.)

[159] L. Wang, Y. Zheng, X. Xie, and W. Ma. A flexible spatio-temporal indexing scheme for large-scale GPS track retrieval. In *Proceedings of the 9th International Conference on Mobile Data Management*, pages 1–8. IEEE, 2008. (Cited on page 36.)

[160] X. Wang, W. Zhang, Y. Zhang, X. Lin, and Z. Huang. Top-k spatial-keyword publish/subscribe over sliding window. *The VLDB Journal*, 26(3):1–26, 2016. (Cited on pages 18 and 25.)

[161] X. Wang, Y. Zhang, W. Zhang, X. Lin, and Z. Huang. SKYPE: Top-k spatial-keyword publish/subscribe over sliding window. In *Proceedings of the VLDB Endowment*, pages 588–599. VLDB Endowment, 2016. (Cited on pages 18, 23, 25, and 99.)

[162] R. T. Whitman, M. B. Park, S. M. Ambrose, and E. G. Hoel. Spatial indexing and analytics on Hadoop. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 73–82. ACM, 2014. (Cited on page 24.)

[163] D. Wu, G. Cong, and C. S. Jensen. A framework for efficient spatial web object retrieval. *The VLDB Journal*, 21(6):797–822, 2012. (Cited on pages 12 and 15.)

[164] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. Joint top-k spatial keyword query processing. *IEEE Transactions on Knowledge and Data Engineering*, 24(10): 1889–1903, 2012. (Cited on page 12.)

[165] D. Wu, M. L. Yiu, and C. S. Jensen. Moving spatial keyword queries: Formulation, methods, and analysis. *ACM Transactions on Database Systems*, 38(1):7, 2013. (Cited on pages 18 and 22.)

[166] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM Transactions on Database Systems*, 36(3):15, 2011. (Cited on page 30.)

[167] H. Yan, S. Ding, and T. Suel. Inverted index compression and query processing with optimized document ordering. In *Proceedings of the 18th International Conference on World Wide Web*, pages 401–410. ACM, 2009. (Cited on page 15.)

[168] Z. Yang, K. Cai, J. Tang, L. Zhang, Z. Su, and J. Li. Social context summarization. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 255–264. ACM, 2011. (Cited on page 100.)

[169] Z. Yin, L. Cao, J. Han, J. Luo, and T. S. Huang. Diversified trajectory pattern ranking in geo-tagged social media. In *Proceedings of the Eleventh SIAM International Conference on Data Mining*, pages 980–991, 2011. (Cited on pages 112, 113, and 117.)

[170] J. Yu, J. Wu, and M. Sarwat. GeoSpark: A cluster computing framework for processing large-scale spatial data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 70:1–70:4. ACM, 2015. (Cited on page 24.)

[171] J. Yu, J. Wu, and M. Sarwat. A demonstration of GeoSpark: A cluster computing framework for processing big spatial data. In *Proceedings of the IEEE 32nd International Conference on Data Engineering*, pages 1410–1413. IEEE, 2016. (Cited on page 24.)

[172] C. Zhang, Y. Zhang, W. Zhang, X. Lin, M. A. Cheema, and X. Wang. Diversified spatial keyword search on road networks. In *Proceedings of the 17th International Conference on Extending Database Technology*, pages 367–378, 2014. (Cited on pages 18 and 28.)

[173] C. Zhang, Y. Zhang, W. Zhang, and X. Lin. Inverted linear quadtree: Efficient top k spatial keyword search. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1706–1721, 2016. (Cited on page 12.)

[174] D. Zhang, Y. M. Chee, A. Mondal, A. K. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *Proceedings of the IEEE 25th International Conference on Data Engineering*, pages 688–699. IEEE, 2009. (Cited on pages 3, 18, 19, and 113.)

[175] D. Zhang, K.-L. Tan, and A. K. Tung. Scalable top-k spatial keyword search. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 359–370, 2013. (Cited on pages 12, 16, 61, 62, 70, 115, and 131.)

[176] D. Zhang, C.-Y. Chan, and K.-L. Tan. Processing spatial keyword query as a top-k aggregation query. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 355–364. ACM, 2014. (Cited on pages 12, 16, 61, 62, 70, and 101.)

[177] M. Zhang, H. Wang, Y. Lu, T. Li, Y. Guang, C. Liu, E. Edrosa, H. Li, and N. Rishe. TerraFly GeoCloud: An online spatial data analysis and visualization system. *ACM Transactions on Intelligent Systems and Technology*, (3):34, 2015. (Cited on pages 25 and 151.)

[178] Y. Zheng. Trajectory data mining: An overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3):29, 2015. (Cited on page 33.)

[179] Y. Zheng and X. Zhou. *Computing with spatial trajectories*. Springer Science & Business Media, 2011. (Cited on page 33.)

[180] Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th International Conference on World Wide Web*, pages 791–800. ACM, 2009. (Cited on page 117.)

[181] Y.-T. Zheng, Z.-J. Zha, and T.-S. Chua. Mining travel patterns from geotagged photos. *ACM Transactions on Intelligent Systems and Technology*, 3(3):56, 2012. (Cited on pages 112, 113, and 117.)

[182] P. Zhou, D. Zhang, B. Salzberg, G. Cooperman, and G. Kollios. Close pair queries in moving object databases. In *Proceedings of the 13th ACM International Workshop on Geographic Information Systems*, pages 2–11. ACM, 2005. (Cited on page 36.)

[183] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. Hybrid index structures for location-based web search. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 155–162. ACM, 2005. (Cited on pages 12 and 15.)

[184] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, 23(4):453–490, 1998. (Cited on page 14.)

Zusammenfassung

Die allgegenwärtige Nutzung von GPS-fähigen mobilen Endgeräten und sozialen Netzwerken führt zu einem immer größer werdenden Volumen an sogenannten *räumlich-textlichen Daten* (z.B. georeferenzierte Beiträge auf Twitter oder Restaurantbewertungen auf Foursquare). Einhergehend mit diesem Anstieg nimmt zugleich die Nachfrage nach Daten mit räumlichen Bezug (z.B. Internet-Suchen nach lokal relevante Informationen) zu. In der wissenschaftlichen Literatur werden Anfragen, wo das Suchkriterium aus textlichen und räumlichen Prädikaten besteht, als Schlüsselwort-Anfragen mit räumlichen Bezug (*spatial keyword queries*) bezeichnet.

Die Literatur beschäftigt sich mit verschiedenen Typen von spatial keyword queries. Diese reichen von einfachen top-k Suchen bis zu komplexeren Anfragevarianten. Die überwiegende Mehrheit der Forschungsarbeiten fokussiert sich allerdings auf die Anfragebearbeitung in rein statischen Szenarien, d.h. die zugrunde liegenden Daten sind eher statischer Natur. In starkem Kontrast dazu steht die Dynamik der sozialen Netzwerke, die kontinuierlich eine große Menge von sich ständig verändernden, nutzer-generierten räumlich-textlichen Daten anbieten. Gerade die Einbeziehung dieser Eigenschaften in die Anfragebearbeitung ist weniger gut erforscht und bietet Raum zur Verbesserung existierender Ansätze.

In der vorliegenden Arbeit beschäftige ich mich daher mit neuen Ansätzen zur Informationsgewinnung und Analyse von georeferenzierten Kommentaren. Zur Einbeziehung der Dynamik erweitere ich zunächst Zugriffsmethoden für spatial keyword queries um eine zeitliche Komponente. Ich betrachte hierbei zuerst Techniken zur Indizierung und Filterung von Trajektorien aus Kommentaren in sozialen Netzwerken. Aufbauend darauf betrachte ich, durch Auffindung einer selektiven Untermenge möglichst repräsentativer Ergebnisse, Ansätze zur explorativen Analyse von großen Datenmengen, die durch eine räumlich-zeitliche Bereichsabfrage mit Schlüsselwortfilter gewonnen werden. Jedoch werden die oben beschriebenen Anfragearten der Dynamik in sozialen Medien noch nicht vollumfänglich gerecht, da die Ergebnismengen durch den kontinuierlichen Strom an neuen Daten schnell veralten. Ich betrachte daher wie vorgenannte Ansätze zu einer Datenstromanalyse erweitert werden können, indem ich Methoden für die kontinuierliche Zusammenfassung von Kommentaren untersuche. Abschließend analysiere ich mit Hilfe von zwei Data-Mining Verfahren den nutzergenerierten Charakter von Kommentaren in sozialen Netzwerken. Hier beschreibe ich zunächst ein System zur Auffindung und Exploration von lokalen Anziehungspunkten an denen bestimmte Schlüsselwörter signifikant häufiger auftreten (*locally trending topics*). Ferner untersuche ich einen Ansatz, der auf der Grundlage von digitalen Spuren von mobilen Nutzern in sozialen Netzwerken thematische Zusammenhänge zwischen verschiedenen Orten auffinden kann.

Erklärung

Ich versichere hiermit, dass ich die vorliegende Dissertation selbständig verfasst habe und alle Hilfsmittel und Hilfen als solche gekennzeichnet sind. Die Arbeit wurde bei keiner anderen Prüfungsbehörde eingereicht.

Paras Mehta

Berlin, den 01.02.2018