DISSERTATION

submitted

to the

Combined Faculty for the Natural Sciences and Mathematics

of

Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

M.Sc. Hamed "Moh'd Tayseer" Abdelhaq

Born in: Kuwait

Oral examination:.....

Localized Events in Social Media Streams: Detection, Tracking, and Recommendation

Advisor: Prof. Dr. Michael Gertz

Abstract

From the recent proliferation of social media channels to the immense amount of user-generated content, an increasing interest in social media mining is currently being witnessed. Messages continuously posted via these channels report a broad range of topics from daily life to global and local events. As a consequence, this has opened new opportunities for mining event information crucial in many application domains, especially in increasing the situational awareness in critical scenarios. Interestingly, many of these messages are enriched with location information, due to the widespread of mobile devices and the recent advancements of today's location acquisition techniques. This enables location-aware event mining, i.e., the detection and tracking of *localized events*.

In this thesis, we propose novel frameworks and models that digest social media content for localized event detection, tracking, and recommendation. We first develop KeyPicker, a framework to extract and score event-related keywords in an online fashion, accounting for high levels of noise, temporal heterogeneity and outliers in the data. Then, LocEvent is proposed to incrementally detect and track events using a 4-stage procedure. That is, LocEvent receives the keywords extracted by KeyPicker, identifies local keywords, spatially clusters them, and finally scores the generated clusters. For each detected event, a set of descriptive keywords, a location, and a time interval are estimated at a fine-grained resolution. In addition to the sparsity of geo-tagged messages, people sometimes post about events far away from an event's location. Such spatial problems are handled by novel spatial regularization techniques, namely, graph- and gazetteer-based regularization. To ensure scalability, we utilize a hierarchical spatial index in addition to a multi-stage filtering procedure that gradually suppresses noisy words and considers only event-related ones for complex spatial computations.

As for recommendation applications, we propose an event recommender system built upon model-based collaborative filtering. Our model is able to suggest events to users, taking into account a number of contextual features including the social links between users, the topical similarities of events, and the spatio-temporal proximity between users and events. To realize this model, we employ and adapt matrix factorization, which allows for uncovering latent user-event patterns. Our proposed features contribute to directing the learning process towards recommendations that better suit the taste of users, in particular when new users have very sparse (or even no) event attendance history.

To evaluate the effectiveness and efficiency of our proposed approaches, extensive comparative experiments are conducted using datasets collected from social media channels. Our analysis of the experimental results reveals the superiority and advantages of our frameworks over existing methods in terms of the relevancy and precision of the obtained results.

Zusammenfassung

Motiviert durch die jüngste Ausbreitung sozialer Medien und der gewaltigen Menge an Nutzer-generierten Daten lässt sich aktuell ein steigendes Interesse im Bereich des *social media mining* beobachten. Die Nachrichten, die kontinuierlich über die Kanäle sozialer Medien verschickt werden, decken eine große Breite an Themen ab – von Informationen über alltägliche Dinge bis hin zu Informationen über lokale und globale Events. Somit entstanden in der jüngeren Vergangenheit ganz neue Möglichkeiten, in verschiedensten Anwendungsdomänen Informationen über Events zu entdecken, vor allem um das Situationsbewusstsein in kritischen Szenarien zu erhöhen. Da mobile Endgeräte immer weiter verbreitet sind und heutzutage standardmäßig Standortdaten versenden können, sind viele dieser Nachrichten mit Ortsinformationen angereichert. Dies ermöglicht ein ortssensitives Mining von Events, das heißt die Erkennung und Weiterverfolgung von lokalisierten Events.

In der vorliegenden Arbeit werden neue Rahmenwerke und Modelle vorgeschlagen, die die Erkennung, Weiterverfolgung und Empfehlung von lokalisierten Events aus den Inhalten sozialer Medien ermöglichen. Zunächst wird das Tool KeyPicker entwickelt. Dabei handelt es sich um ein Rahmenwerk, mit dem Event-relevante Schlüsselwörter während der Verarbeitung extrahiert und bewertet werden können, wobei dem hohen Maß Rauschen, der zeitlichen Heterogenität sowie Ausreißern in den Daten Rechnung getragen wird. Danach wird LocEvent vorgestellt, um basierend auf einer vierstufigen Methode Events zu erkennen und weiterzuverfolgen. Somit erhält LocEvent als Eingabe die durch KeyPicker extrahierten Schlüsselwörter, identifiziert daraus die lokal-relevanten Wörter, gruppiert diese basierend auf räumlichen Informationen und bewertet schließlich die erstellten Gruppen. Für jedes extrahierte Event werden somit eine ausdrucksstarke Menge an Schlüsselwörtern, ein Ort sowie ein Zeitintervall bestimmt. Neben der geringen Menge an Nachrichten mit Ortsinformationen im Vergleich zur Gesamtmenge an Nachrichten ist dabei auch zu berücksichtigen, dass manche Nutzer auch über von ihnen weit entfernte Events schreiben. Diese raumbezogenen Probleme werden mithilfe neuer Regularisierungstechniken adressiert, die Graphen- und Gazetteer-basiert arbeiten. Um die Skalierbarkart zu gewährleisten wird dabei zusätzlich zu einem hierarchischen räumlichen Index ein mehrstufiger Filterungsprozess verwendet, der nach und nach unpassende Wörter filtert und nur Event-relevante Wörter für die komplexen räumlichen Berechnungen berücksichtigt.

Desweiteren wird ein Empfehlungssystem für Events vorgeschlagen, das, wie aktuelle Empfehlungssysteme auch, auf einem *collaborative filtering* Ansatz basiert.

Mithilfe des Models können Nutzern Events vorgeschlagen werden, wobei eine Menge an Kontextinformationen berücksichtigt werden, wie z.B. soziale Verbindungen zwischen Nutzern, die thematische Ähnlichkeit von Events sowie die räumliche und zeitliche Nähe zwischen Nutzern und Events. Um das Modell zu realisieren, wird die Methode der Matrixfaktorisierung verwendet und angepasst. Das ermöglicht versteckte Nutzer-Event-Muster aufzudecken. Die für das Modell verwendeten Features tragen dazu bei, den Lernprozess hin zu Empfehlungen von Events zu steuern, die dem Interessen des Nutzers am nächsten kommen. Dies ist vorallem hilfreich, wenn bei neuen Nutzern wenige oder gar keine Informationen vorliegen, an welchen Events bereits teilgenommen wurde.

Um die Effektivität und Effizienz unserer vorgeschlagenen Ansätze zu evaluieren, werden umfangreiche Experimente durchgeführt, wobei auf Datensätze verschiedener sozialer Medienkanäle zurückgegriffen wird. Die Analyse der Evaluierungsergebnisse belegt dabei die Überlegenheit und die Vorteile unserer Frameworks gegenüber existierenden Methoden bezüglich der Relevanz und Genauigkeit der erzielten Ergebnisse.

Acknowledgements

Undertaking this PhD has been a truly life-changing experience for me. This work would not have been possible without the support that I received from many people. First of all, I would like to express my deepest gratitude to my advisor, Prof. Dr. Michael Gertz, for giving me the opportunity to be one of his PhD students, it is truly an honor. I am really thankful for his guidance, endless support, immense knowledge, and deep insights that helped me at various stages of my research. I remain amazed that despite his busy schedule, he was able to go through the final draft of my thesis and meet me regularly with comments and suggestions on almost every page. I would also like to thank my dissertation committee for the time, efforts, and precious feedback.

During my PhD study and in spite of my busy days, I had a memorable time at the Database Systems Research Group, Heidelberg University. I was lucky to have an impressive research environment with great colleagues. Thank you, Dr. Ayser Armiti, you supported my first step to join this wonderful group. Many thanks to Dr. Tran Van Canh, Florian Flatow, and Katarina Gavrić for all the useful discussions, brainstorming sessions, and fun we had together in our office. Also, many thanks to all further group members: Dr. Jannik Strötgen, Dr. Christian Sengstock, Dr. Le Van Quoc Anh, Thomas Bögel, Hui Li, Dr. Johanna Geiß and Andreas Spitz for your friendship and support over the years. I would also like to thank all of my friends who assisted and incited me to strive towards my goal.

I would like to express my sincere gratitude to the "Deutsche Akademische Austauschdienst Dienst" (DAAD) for the financial support granted through their scholarship program. Without this scholarship, it would not be easy for me to have this work done.

Words cannot express how grateful I am to my mother and my father for all of the sacrifices that you've made on my behalf. Your prayer was what sustained me thus far. My warm thanks to my brothers and sisters for the spiritual support throughout my life and PhD years.

Last but not least, I want to thank my beloved wife Bayan Takruri for her patience, assistance, support and faith in me, which was, at the end, what made this dissertation possible. Bayan, you helped me to regain hope after despair, restart journeys after detours, revive strength after defeat and resurrect dreams after rejection. Thanks to my mother-in law and father-in-law for raising such a great daughter. The long hours away from you and our source of happiness, our sons, Tayseer and Kareem, during my PhD study were truly difficult.

To my parents, my wife, and children.

Contents

Chapter 1

Introduction

The recent and rapid advancements of social media sites such as Twitter[1] and Facebook[2] have provided virtual and easy-to-use channels for users to communicate, to report their thoughts, and to share what is happening. To date, a large number of such social media services are being accessed by hundreds of millions of users who publish content on a daily basis. Microblogging services such as Twitter are among the leading services causing this dramatic increase in user-generated content [85]. Users constantly publish brief textual updates, coined "microblogs", describing their daily activities and surrounding phenomena. As of September 2015, about 316 million monthly active users publish microblogs using Twitter[3]. To better imagine how large this number is, it is slightly larger than the population of Indonesia (252 million) and more than 3 times larger than that of Germany (80 million). For Facebook, the current statistics are even more impressive, with about 1.35 billion active users per month[4].

Social media break the barriers between the real world and the virtual world, facilitating new opportunities to understand individuals and to extract actionable patterns from this content [137, 165]. Consequently, social media mining has recently received considerable attention across various research areas, including data mining, spatio-temporal data analysis, machine learning, and statistics, to name but a few. Knowledge and meaningful patterns extracted from social media can be utilized in many application domains such as *event detection* [4, 35, 130, 151], *social opinion analysis* [86, 145], and *recommender systems* [36, 117].

[1] `https://twitter.com/`

[2] `https://www.facebook.com`

[3] `https://about.twitter.com/company`, accessed Sept. 2015

[4] `http://blog.lewispr.com/2014/12/the-most-recent-social-media-statistics.html`, accessed Feb. 2015

Among the vast variety of applications in the field of social media mining, we focus on studying the spatio-temporal characteristics of event-related words towards detecting, tracking, and recommending real-world events. We first present the motivation and the research objectives of our study in Section 1.1. Then, the key challenges faced in achieving these goals are discussed in Section 1.2. The main contributions of this thesis are summarized in Section 1.3 before concluding this chapter with the thesis outline in Section 1.4.

1.1 Motivation and Research Objectives

The huge amount of content published via microblogging services can be viewed as an up-to-date source of event-related information [30, 136, 156, 171]. Users participating in or observing an event are motivated to publish a relatively large number of related microblogs, and thus act as *human sensors*. The semantic dimension of this content that is continuously published via these sensors adds great value to the event information. Research has primarily focused on identifying and tracking events from social media using the temporal context of messages [8, 28, 156]. Nowadays, however, the availability of several location-acquisition systems, e.g., Global Positioning System (GPS) or Radio-Frequency Identification (RFID), has led to enriching the stream of microblogs with geographic information that is utilized either as place names embedded within the text of microblogs or as geo-coordinates attached to these messages. As a result, it becomes possible to study and detect another type of events having a spatially-limited spread, namely, *localized events*. Localized events, e.g., festivals and musical concerts, are real-world events stimulating people to gather at a specific location for a limited period of time in order to participate, attend, or even observe what is going on. Monitoring localized events over social data streams has many applications. Some of those applications are outlined below.

- *Crisis Management.* Collecting event information from social media has a great impact on increasing the overall situational awareness during crises [56, 130]. For example, at the height of 2011 South East Queensland floods during 10th - 16th of January, more than 35,000 microblogs containing the hashtag #qldfloods were sent using Twitter. These event-related microblogs covered different important aspects of the crisis, such as reporting a sinking yacht in Brisbane river and observing a bull shark swimming past the McDonald's restaurant in

Goodna[1] etc. This real-time information helps first-aid officers to make suitable responses on time and hence mitigate the consequences of such critical events.

- *Decision Making.* Many services can benefit from the up-to-date event information extracted from social media. For example, the real-time detection of a road jam helps navigation systems to provide more reliable routing information in such a situation.

- *Event Recommendation.* User-contributed content published on a certain event might give a better representation of previous instances of this event from an attendee's perspective [25], or might provide immediate relevant information such as "tickets for Apple shop opening are sold out". This event-centric information is an important input to recommender systems to improve event recommendations.

Motivated by a vast variety of application domains and the vital role that event detection plays, when conducted in a timely manner, we devote our efforts in this thesis to studying the problem of localized event detection, tracking, and recommendation, which will be defined in more detail below.

Localized event detection and tracking

We build a framework in support of detecting and tracking localized events in an online fashion using the sliding window model. As the window slides and new microblogs enter the system, the framework incrementally performs the following fundamental tasks: (1) extracting event-related words, called *keywords*, and associating each with a dynamic score reflecting its change in significance over time. (2) Identifying keywords having a spatially limited extent, i.e., *local keywords*. (3) Using these local keywords to form new event groups (clusters) or to update existing event clusters. (4) Scoring each detected localized event in order to quantify its evolution over time. In addition, for each detected event, a set of ranked keywords, a fine-grained geographic location, start time, and end time are estimated.

Event recommendation

As an application, we propose a model for personalized event recommendation based on the model-based collaborative filtering paradigm [89, 123]. A ranked list of future

[1] `http://www.qt.com.au/news/ipswich-bull-sharks-spotted-flood-affected-streets/743873/`, accessed Feb. 2015

events are suggested to a user, taking into account three contextual features: social, topical, and spatio-temporal features. One major objective in this work is to exploit such features in order to build a recommender system that can deal with the *cold-start* problem, i.e., when recommending events to a new user who has no (or a very sparse) activity history.

1.2 Key Challenges

The efficient extraction of reliable patterns from the wealth of event data that is buried in the content of social media entails adopting non-traditional methodologies to tackle several new challenges:

(1) *Noisy and dynamic content.* The textual content published via social media sites contains a lot of typos, colloquial words, shortcuts, application-specific tokens (e.g., hashtags in Twitter) etc. On the other hand, at every single moment, a large number of new tokens that might hint to some new trending topics appear. Existing event mining techniques assuming a predefined set of vocabularies fail to easily and efficiently subsume this noisy and dynamic content.

(2) *Temporal heterogeneity and outliers.* As for temporal heterogeneity, the usage patterns of the majority of words vary over time, leading to mistakenly identify words as event-related keywords. For example, the sudden surge of word "work" at 09:00 am does not mean that a work-relevant event has occurred, yet it is a daily habit of people to talk about their work in the early morning. On the other hand, people tend to post about events before/after they start/end, which, for example, might prevent estimating accurate time intervals of events. This latter phenomenon is called temporal outliers.

(3) *Spatial sparsity and outliers.* The sparse distribution of event-related microblogs over space, in particular in low-populated regions, and the fact that these microblogs are published from locations distant from the actual locations of their respective events are spatial problems that we refer to as spatial sparsity and outliers, respectively. Both problems prevent an accurate estimation of the location of an event.

(4) *Volume of incoming microblogs.* In Twitter, for instance, about 6000 microblogs/second are posted by users all over the globe. Hence, any practically

relevant framework needs to be highly scalable in order to digest this huge data stream and provide near real-time event information.

(5) *Difficult evaluation*: A standard method for evaluation in data mining is to compare extracted patterns with some given ground truth. Unfortunately, ground truth is often unavailable in the context of social media mining, making a standardized evaluation process almost impossible. Therefore, testing the validity of extracted event patterns is non-trivial and mainly depends on domain knowledge.

1.3 Contributions

This thesis includes several contributions in the field of event detection, tracking, and recommendation in social media:

(1) We propose a 4-stage framework tailored to the online detection and tracking of localized events from social media. This framework is based on the feature-pivot paradigm (see Chapter 2) where, first, a set of keywords is extracted from huge volumes of noisy terms. The spatial distributions of such keywords are then analyzed to identify local ones. After that, local keywords are spatially grouped to generate potential event clusters that are finally scored to capture their evolution over time. We estimate for each detected event a set of descriptive keywords, its start time, and spatial location.

(2) Since event-related keywords are the main building blocks in feature-pivot detection, we analyze their spatio-temporal distributions and develop effective solutions for a number of temporal and spatial problems. The concepts of *time-aware baseline* and *word recurrence* are introduced to handle temporal heterogeneity and temporal outliers, respectively. To mitigate the adverse impact of spatial outliers, we propose two novel signature regularization techniques: graph- and gazetteer-based regularization. Finally, a non-parametric kernel density estimate is employed to smoothen the spatial signals of keywords over space and thus to cope with spatial sparsity.

(3) Identifying local keywords and estimating their (central) locations in a fine-grained spatial resolution are computationally expensive tasks, yet essential to determine the locations of localized events. For this, besides applying the sliding window model to process only the most recent microblogs, we utilize a

hierarchical space-partitioning index to ensure a fast pruning of the geographic space and thus a scalable system. Furthermore, synopses (summaries) of the spatial component of incoming microblogs are maintained within this index structure that is updated constantly as the time window slides.

(4) We propose a context-aware event recommendation system that accounts for social, topical, and spatio-temporal features to provide personalized event recommendations. Our proposed model is built upon the well-known matrix factorization technique that is one of the most successful realizations of model-based collaborative filtering. Matrix factorization shows better performance compared to other methods in the Netflix[1] recommendation challenge in terms of scalability and accuracy. Furthermore, we show that our model is generic as new contextual information can be integrated easily. This is achieved by only updating one or a combination of the above three features that cover all possible interactions between the system's main entities, namely, users and events.

We have three publications in the context of event detection and tracking. First, a proof-of-concept application that demonstrates our localized event detection framework was published in: *EvenTweet: Online Localized Event Detection from Twitter* (VLDB 2013) [4]. Our contributions in handling spatio-temporal problems of event-related keywords were partially published in: *Spatio-temporal Characteristics of Bursty Words in Twitter Streams* (ACMGIS 2013) [3]. An efficient solution to the problem of local keyword identification and spatial focus estimation using a fine spatial granularity was published in: *On the Locality of Keywords in Twitter Streams* (IWGS/ACMGIS 2014) [2]. However, the work presented here extends these in significant ways, including more detailed and alternative model descriptions, more extensive experiments, and new application models.

1.4 Thesis Outline

The remainder of the thesis is organized as follows.

- **Chapter 2.** We first introduce some concepts, challenges, and applications of social media mining. Then, we review the main techniques employed in Topic Detection and Tracking (TDT) to extract event information from traditional media such as newswire sources. Such techniques have been and are adapted

[1] `http://www.netflixprize.com/`, accessed Apr. 2015

to cope with the noisy and large volumes of content published via social media services. Therefore, in this chapter, a large number of related research efforts are discussed and categorized based on the methodology adopted and the objective of the event detection process.

- **Chapter 3.** We propose an incremental approach called KeyPicker to extract keywords from a stream of microblogs. For this, we first present how incoming microblogs are temporally indexed and preprocessed. Then, the concepts of word signal and baseline are introduced to quantify the usage patterns of words and thus to distinguish keywords among other noisy words. The results obtained from the experimental evaluation using Twitter datasets reveal the efficiency and reliability of the proposed approach in extracting and weighting keywords.

- **Chapter 4.** We propose a framework called LocEvent for the online detection and tracking of localized events using the sliding window model. We first present the main stages of LocEvent that starts by spatially processing the keywords extracted by KeyPicker. For this, the concept of spatial signature is introduced and the major spatial problems in estimating sound signatures are highlighted. We then present novel techniques to adjust spatial signatures and show how adjusted signatures are efficiently used to identify local keywords. Local keywords are then spatially clustered and the generated clusters are scored based on the temporal characteristics of the contained keywords. We conducted extensive experiments using Twitter data, and the results are evaluated to show the efficiency, utility, and advantages of LocEvent compared to others.

- **Chapter 5.** In this chapter, we present our proposed context-aware event recommender system on the basis of model-based collaborative filtering, realized using matrix factorization. We first describe our proposed contextual features: social, topical, and spatio-temporal features and then show how to integrate these features into the collaborative filtering model towards better event recommendations. Using a dataset from Meetup platform, we evaluate the robustness of our model and compare it against a number of alternatives where one or more contextual features are not taken into account.

- **Chapter 6.** Finally, we summarize our work in this dissertation and present open issues for further studies.

Chapter 2

Background and Related Work

2.1 Overview and Objectives

As discussed in the previous chapter, this work aims at extracting spatio-temporal event patterns from social media in an online fashion, which are then utilized in building a spatio-temporal-aware event detection and tracking framework. What distinguishes our work from existing studies is that it gives more attention to the spatial dimension of content, towards the detection and tracking of localized events using a fine-grained spatio-temporal resolution. For this, techniques for spatio-temporal data analysis, stream processing, natural language processing, and indexing are exploited to build effective and efficient models and algorithms. We begin this chapter by briefly introducing general concepts, challenges, and applications of social media (Sections 2.2). Then, in Section 2.3, we present the first serious attempts to detect and track events from text streams, which started by the Topic Detection and Tracking Project (TDT). Concepts and models related to data streams and stream processing are addressed in Section 2.4. After that, the main approaches on event detection in traditional media, e.g., newswire sources, are presented in Section 2.5. Then, we show how these approaches have evolved and been refined to cope with the new challenging and noisy nature of social media content (Section 2.6). The most related are categorized here based on the methodology used and the objective of the study. Finally, in Section 2.7, we conclude this chapter and discuss the main issues still open in the field of event detection and tracking.

2.2 Social Media

The rise of Web 2.0[1] technology has enabled users to generate their own content, publish it, and interact with each other. This explains the increasing prevalence and popularity of social media services nowadays, offering new virtual communication channels for people to interact and to disseminate information that they likely would not share otherwise using traditional channels, e.g., by email, phone, and IM [85, 169].

To date, a large number of social media services have been developed to fulfill the social needs in sharing content and interacting with each other over the Internet. Such services can be divided into several categories including, but not limited to, social networking (Facebook and LinkedIn), microblogging services (Twitter and Foursquare), photo sharing (Flickr and Instagram), video sharing (YouTube), news aggregation (Google reader and Feedburner), social search (Google, Ask.com), to name but a few [165]. Kaplan and Haenlein [88] defined *social media* as "a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of user generated content". On the other hand, people emotionally seem to use social media to achieve a level of virtual connectivity with friends and the world [115].

2.2.1 Social Media Mining

Social media contributes to removing the boundaries between the real and the virtual world by allowing users to publish very specific details about their activities, thoughts, reactions towards natural phenomena etc., resulting in an amazingly tremendous volume of content being steadily published. This content hides a wealth of valuable information to be extracted and analyzed. Therefore, social media mining can be defined as the process of representing, analyzing, and extracting actionable patterns from social media data [165]. This research area introduces basic concepts and algorithms adapted for analyzing massive social media data. Different disciplines, such as data mining, machine learning, statistics, optimization, and graph theory are employed to come up with models and techniques to process and reliably analyze the content of social media.

However, mining useful patterns from the content of social media leads to a number of new challenges, which can be summarized as follows:

[1] `http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html`, accessed Mar. 2015

1. **Immense volume of data**: Nowadays, a large number of messages are being continuously published via several social media services. For example, from Facebook alone, about 115,740 text updates are posted per second all around the globe. To extract hidden knowledge from this huge amount of content, traditional batch-based processing techniques perform poorly and prevent a reasonable response time.

2. **Noisy content**: Noise is defined as a random error or variance in a measured variable [73, 158]. However, social media suffers from another sort of noisiness, namely, *textual noise*. That is, the content of social media has a lot of typos, colloquial terms, abbreviations, phonetic substitutions, ungrammatical structures etc. Therefore, data preprocessing and noise removal is inevitable in the context of social media analysis.

3. **Sparse data**: Usually, in social media, informative observations that can be linked to a certain interesting pattern or phenomenon to be detected and analyzed are infrequent. To mitigate this sparsity, one needs to aggregate observations over a relatively long period of time to obtain statistically significant and representative results. Moreover, one of the commonly used methods for retrieving data from social media services is via application programming interfaces (APIs). Using these APIs, only a limited rate of the actual stream is available, which exaggerates the problem of data sparsity in social media [3, 24].

2.2.2 Applications of Social Media Mining

The prevalence of user-generated content via social media services has encouraged the development of a wide range of applications. Examples of such applications are:

Detecting real-world events

The main stimuli that encourage people to share information via social media services are real-world events, in particular, natural hazards, such as earthquakes and floods [4, 35, 130, 151]. The immediate detection of these events from the content of social media helps mitigate their adverse consequences and increase the overall situational awareness. In this thesis, we focus on identifying event-related keywords that exhibit a surge in usage and analyze their spatio-temporal behavior, which can then be used in detecting and tracking localized real-world events. For further details on event detection and tracking from social media, see Section 2.6.

Social opinion mining

The popularity of social media has fueled the need for opinion mining from user-generated content [86, 145]. Enterprises compete nowadays to utilize natural language processing and sentiment analysis techniques to understand the conversations, to identify relevant content, and to know whether people like or dislike their products/services, which, in turn, helps them manage their reputation accordingly.

Recommender systems

These systems represent a wide category of Web applications that involve predicting user responses to options [24, 58], e.g., product and movie recommendations. Of course, the content of social media provides a context-aware and semantically-rich environment when utilizing the underlying link structure and attached spatio-temporal information [126]. Hence, social media is helpful in improving the results obtained from traditional recommendation systems.

2.2.3 Microblogging Services

One of the most popular category of social media platforms are microblogging services such as Twitter, Tumblr[1], and Foursquare[2], which enable users to broadcast brief text updates (called *microblogs*) about things occurring in their daily life, work activities etc. For example, the users of Twitter share these microblogs with friends, families, co-workers, or even with the public [169]. The increasing popularity of such microblogging services makes them a strategic source of valuable information. In the following, we give a more detailed description of Twitter as it is our main source for datasets used throughout this thesis.

Twitter

Twitter is one of the most popular and fast-growing microblogging services, where users all around the world publish short messages called *tweets*. At this time, about 500 million tweets, written in more than 35 languages, are posted per day via the Twitter website, SMS, and mobile applications[3]. The majority of Twitter active users (about 80%) publish their tweets using their mobile applications. Consequently, the Twitter website is ranked among the top 10 most popular sites according to

[1] `https://www.tumblr.com/`
[2] `https://foursquare.com/`
[3] `https://about.twitter.com/company`, accessed Mar. 2015

Figure 2.1: A sample tweet shared from account (BBCClick). The token "#CERN" is a hashtag describing the topic of the tweet, and "@engadget" is called a mention and used to draw the attention of user "engadget" to the topic of this tweet. .

Alexa traffic ranking[1]. A key characteristic of tweets is that they are limited to 140 characters and are much more concise than blog posts, allowing for effective and timely dissemination of information over the Internet. Twitter is different from other social media services by the following/follower relationship. If, for example, user $u1$ follows user $u2$, then $u1$ becomes a follower to $u2$ even without an approval from $u2$, and as a result $u1$ can view the tweets posted by $u2$. In contrast, $u2$ cannot view the tweets of $u1$ unless he/she follows $u1$.

Twitter currently provides a number of markup symbols, each of which has a specific role (see Figure 2.1). For example, the "@" symbol that is placed before a user name ("@username") forms a *mention*. If a tweet contains a mention of a certain user, this means that the sender wants to draw his/her attention to the topic of the tweet [79]. When mentions are placed at the very beginning of a tweet, then it is called a "reply", indicating that the sender intended to respond to a tweet of the mentioned user. The hashtag symbol "#" is utilized to attach some key phrases to a tweet. Tweets having one or more common hashtags are assumed to be posted on the same topic, i.e., the topic inferred from these hashtags. This allows users to regularly track specific trends or events in real-time.

The content of Twitter messages covers a broad range of topics. Nevertheless, the majority of published tweets do not contain valuable information. According to a study accomplished by Pear Analytics [1], about 50% of the tweets are either pointless babbles, self promotions, or spams. For this study, 2,000 tweets were manually classified into six categories, as can be seen in Figure 2.2.

Twitter provides an application programming interface (API[2]) that allows for accessing public tweets filtered by location, keyword, and author. The availability of Twitter data collected using this API has attracted the attention of researchers to start

[1] `http://www.alexa.com/siteinfo/twitter.com`, accessed Mar. 2015
[2] `https://dev.twitter.com/streaming/overview`, accessed Mar. 2015

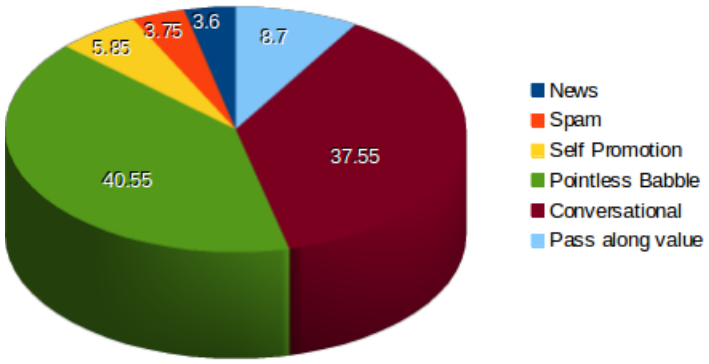


Figure 2.2: Classification of the content of tweets according to Pear Analytics [1].

analyzing this huge noisy content and encouraged industry towards the development of a large number of applications and tools on top of this content.

2.3 Topic Detection and Tracking

Topic Detection and Tracking (TDT) is a name given to a project that is considered to be the first serious attempt towards the development of new technologies with the aim of extracting events from news-oriented textual materials. The extracted events, in turn, are utilized for conducting an event-centric organization of these news stories accordingly. In this section, we start by discussing the motivation behind initiating such a project (Section 2.3.1). Then, we present the main tasks of the TDT project (Section 2.3.2), and finally, some main concepts (event, topic, and trends) are described by showing the relationship between them and how their definitions have evolved over time.

2.3.1 Query-free Information Retrieval

During the last two decades, a rapidly increasing amount of electronically available information has led to the appearance of new challenges for information retrieval (IR) technologies. A new type of queries has arisen, answering more generic questions such as “what happened?” or “what is going on right now?”. These questions cannot be answered using traditional query-driven IR techniques. Assume, for example, a person who has just returned back home from work and wants to know quickly what events are currently taking place. Navigating a number of news Web portals and reading through several articles is a tedious task that needs much time and effort. In addition, generating queries without prior information about ongoing events makes no sense.

On the other hand, if the user has an idea about the event he/she is interested in, the generated query might be either (1) too abstract, resulting in many irrelevant articles (low accuracy), or (2) too specific, which leads to the loss of related articles (low recall) [161].

Motivated by the difficulty in answering general questions by traditional query-based IR technologies, a new research direction that aims at building automatic intelligent systems has been established and described as *query-free* IR. In the context of event mining, such IR paradigm is capable of:

- detecting important events from large volumes of news articles.
- summarizing the content of detected events at different levels of abstraction.
- notifying users immediately as soon as a new event occurs.
- tracking the evolution of events as new articles arrive.

This research direction is called *Topic Detection and Tracking (TDT)* [12, 14]. The name starts with "topic"; however, the goal is to detect and track dynamically changing topics, i.e., events. Later, we will show how the event definition evolves over time and how to distinguish between the two concepts "event" and "topic".

2.3.2 TDT Program

The U.S. government initiated the TDT research in 1996 and supported it for several years. The aim was to develop technologies that search, organize, and conduct event-based structuring of multilingual textual stories arriving from different types of sources, e.g., newswire sources or TV and radio broadcasts. The Linguistic Data Consortium (LDC) [51] transcribed hundreds of TV and radio news recordings, and the corpus was divided into disjoint stories (documents)[1], identifying a *story* as "a topically cohesive segment of news that includes two or more declarative independent clauses about a single event" [61]. For evaluation purposes, LDC prepared annotated corpora, e.g., TDT1, TDT2, and TDT3, as ground truth to evaluate the performance of the approaches proposed to accomplish the tasks defined by the TDT project. The process of Topic Detection and Tracking consists of three main technical tasks [12]:

(1) **The Segmentation Task**: The input to the TDT process is a continuous stream of text, in particular, transcribed speech, that needs to be segmented

[1]The terms "story" and "document" will be used interchangeably in this chapter.

into stories. A key issue here is that the segmentation task should be accurate enough to correctly locate the boundaries between adjacent stories.

(2) **The Detection Task**: Event detection is basically a discovery problem [162], whereby the main objective is to identify events from a stream of documents. Such events are unspecified in advance, and therefore, query-free approaches are required to extract and represent these events. Story clustering appears to be a natural approach for such an event discovery (detection) task. A main assumption is that a story discusses at most one event, and hence, can be assigned to at most one cluster. Based on the technique used to detect events from a stream of documents, event detection is classified into:

- Retrospective Detection (**RED**): Given a corpus of stories, the task is to retrospectively identify all events by forming potential event clusters. However, the timely detection of previously unseen events is not that important in this category.
- Online Detection (**NED**): Given the most recent story from a stream of stories, the aim is to determine whether or not a new event is discussed by this story. If an arriving story is perceived as being discussing a previously unseen event, the story is flagged YES and a new cluster is generated, reporting the occurrence of a new event. Therefore, this task is also termed *first story detection FSD* in the sense that the story flagged YES is the first story discussing the new event. Of course, the posting time of such a story is a good indicator of the start time of its respective event. On the other side, if the arriving story pertains to an existing event, it is flagged NO and assigned to the corresponding event cluster.

(3) **The Tracking Task**: Once a potential event cluster is generated and described by a set of associated stories, the event is consequently defined to be "known". Tracking an existing event means trying to find unlabeled stories that can be assigned to the cluster of this event. Different supervised and unsupervised approaches are used for this task as discussed later in this chapter.

In addition to these primary TDT tasks, there is another secondary task called "the linking task", which mainly tries to answer the question of whether two stories are discussing the same topic, i.e., the stories are "linked" by a common topic [61].

2.3.3 Events versus Topics

An *event* is a fundamental concept in the TDT project. In the pilot study [12], an event is defined as "some unique thing that happens at some point in time", where only the temporal aspect of events is considered. Later, the spatial dimension was included, and hence, an event is defined as "something that happens at some specific time and place" [161] as the majority of events occur at specific locations, despite that some events have global nature, e.g., Mother's Day. However, this definition has a significant shortcoming when an event leads to a number of other events, consequences, and activities. For example, the flooding that took place in Southern Africa on January 1st, 2015, is considered an event. However, additional news stories were published, talking about the relief campaigns sent to the event location, traffic stoppage, the destruction of the electricity system etc. All these resulting events and activities are treated as separate events under the former event definition. To resolve this ambiguity, a new definition was introduced in [11], stating that an event is "a specific thing that happens at a specific time and place along with all necessary preconditions and unavoidable consequences".

Regarding the notion of *topic*, it had a definition similar to the latter event definition. But, later, TDT defines a topic as "a seminal event or activity, along with all directly related events and activities" [11], where the notion *activity* means "a connected set of actions that have a common focus or purpose". For instance, the relief campaigns in our previous example are types of activities for the flooding event in Southern Africa. This topic definition exhibits a scope problem because the word "related" can be interpreted differently without restricting rules. Therefore, the TDT annotation was modified to include the "Rules of Interpretations" (ROI) [11]. The ROI can be seen as a higher-level categorization of events [37]. For example, both a car accident and a motorcycle accident relate to the same "accident" ROI. As a conclusion, an event can be described as an instance of a certain topic. In other words, all related and resulted events and activities fall under the same domain (topic) [60].

2.3.4 Trends

A trending topic (also termed "trend" or "emerging topic") is defined by Kontostathis et al. [91] as "a topic area that is growing in interest and utility over time". That is, a trend may hint to the occurrence of a real-world event that led to the sudden appearance of such a trend and its increasing popularity as well. Detecting trends from news stories provides useful and immediate thematic descriptions for the trig-

gering events. Furthermore, the amount and speed of growth necessary for a topic to be identified as a trend varies based on the data source [25]. For example, trends on Twitter generally change on an hourly basis, while those corresponding to a particular interest on a certain research topic and observed in a number of publications rise and fall less rapidly due to differences in temporal dynamics and the nature of the shared content.

In case of natural hazards, e.g., earthquakes, detecting trends from social media content provides timely and valuable information for news reporter and decision makers in a manner faster than via traditional reporting techniques, and thus helps mitigate different kinds of crises more efficiently.

2.4 Data Streams

Recently a new category of data-intensive applications has become widely recognized, where input data is modeled best as transient data streams and not as persistent relations [19]. Examples of such applications include sensor networks, network monitoring, social media analysis, to name but a few. For these and similar applications, it is not feasible to store, manipulate, and query arriving data using traditional database management systems. This requires revisiting the way by which this data is dealt with, and consequently, has given rise to introducing *Data Stream Management Systems* (e.g., STREAM [16]) that provide good-equipped architectures to handle complex and numerous *continuous queries* over data streams [148].

2.4.1 Data Stream Management Systems

A *data stream* is an ordered sequence of items $(x_1, x_2, \cdots, x_N, \cdots)$ that arrive in a continuous, rapid, and time-varying manner [19]. Figure 2.3 suggests a high-level organization of a Data Stream Management System (DSMS). One or more streams can enter the system. The items arriving within one stream might differ from those of other streams in terms of data types, arrival rates etc. A *streaming algorithm* takes as input a sequence of data items (data stream) such that the sequence is scanned only once (or in only a few passes) in the increasing order of the indexes in order to answer queries on the stream [126].

Although streaming algorithms are designed to process data incrementally (also called *single-pass algorithms*), many query processing and mining algorithms require an efficient execution, which is difficult to achieve with a fast data stream. In ad-

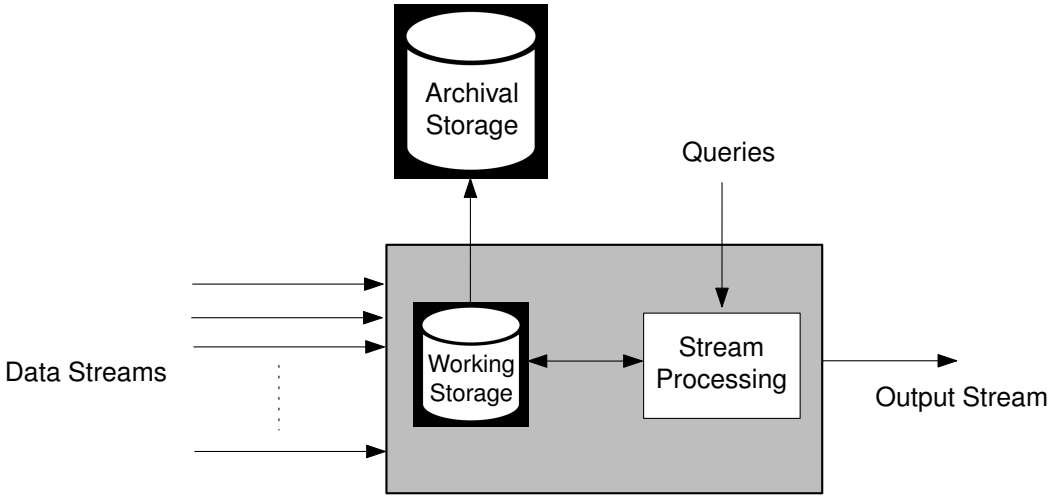


Figure 2.3: Data Stream Management System

dition to the required computational power, another important challenge for DSMSs is to deal with an infinite data stream using a limited amount of memory (denoted working storage, as depicted in Figure 2.3). Generating approximate solutions for different related problems, such as query answering, are acceptable in many application domains. These solutions can be categorized as follows:

(1) **Maintaining synopses of the stream**. *Synopses* are statistics or aggregates that summarize the data items seen so far. These summaries are maintained in some data structures and are used to produce exact or approximate answers to some specified queries (called *standing queries*). For example, suppose that a stream of temperature values is constantly retrieved from a certain sensor. A query that we might be asked is the minimum temperature that is ever recorded by this sensor. To answer this query, we can maintain a single summary that is the minimum value of all stream items ever seen. Although, synopses are very useful in reducing the required space and in efficiently answering queries, they are query-oriented in a sense that different types of queries require different synopses structures. A variety of techniques have been introduced for synopses construction. Some of these techniques are sampling methods, histograms, wavelets, sketches, and micro-cluster-based summarizations [6, 126].

(2) **Window monitoring**. In this line of solutions, only a portion of the streaming items is monitored and kept to answer queries. This is basically motivated by the fact that most recent streaming items are more relevant, whereas old items should have either none or minor impact on a query's answer. Moreover,

window monitoring is more appropriate when asking ad-hoc queries, i.e., arbitrary (unspecified) queries about the stream because the most recent items are maintained themselves rather than synopses of them.

Much of stream data resides at a multi-dimensional space, where each dimension can be thought of as a separate stream. Several examples of such streams have high-arrival rates, e.g, the stream of messages published using Twitter services with a rate of about 6000 tweets/second. Therefore, it might be impossible to fit these huge multi-dimensional streaming items in memory even with monitoring only the messages arrived within a certain time period (window). This, as a result, has given rise to hybrid approaches that entail maintaining synopses over a monitored time window [2, 8].

In social media mining, the line of streaming algorithms is favored over offline ones in digesting and processing the stream of incoming microblogs [7]. In Section 2.5, we elaborate more on streaming approaches that try to leverage from synopses and window monitoring to efficiently detect and track events from social media content.

2.4.2 Window Models

Two categories of data stream monitoring can be identified: (1) point monitoring and (2) window monitoring [172]. In point monitoring, only the latest data item in a stream is considered. If this item fulfills some predefined condition, an action is triggered accordingly. For example, we might have a query to trigger an alert whenever the temperature exceeds 23 degree Celsius. Point monitoring does not need much implementation efforts since only the most recent data item is considered. On the other hand, window monitoring requires maintaining portions of the streaming items (or synopses of them) based on certain time intervals (windows). The three well-known window models that are adopted for stream processing are:

(1) **Landmark windows**. Synopses are calculated based on values observed between a specific time point (i.e., a landmark) and the current time.

(2) **Sliding windows**. Here, synopses are computed based on the last c values in the data stream, where c is a predefined size of the sliding window. For example, the average stock price of Microsoft during the last 10 days is calculated based on a sliding window model.

(3) **Damped windows**. According to this model, the weights of data items decrease exponentially into the past. Assume, for example, a new data item x has

arrived, the new average $average_{new}$ is computed as follows:

$$average_{new} := p \times average_{old} + (1 - p)\ x, \quad where\ (0 < p < 1). \tag{2.1}$$

The sliding window model is the most popular among these window models as it prevents stale data from influencing the maintained synopses and the quality of the results. This model also acts as a tool for approximation in the presence of bounded memory [19]. Several research efforts have been conducted in extending summarizations techniques (synopses construction) to sliding window models, see, e.g., [54, 66].

2.5 Event Detection from Traditional Media

A broad range of techniques have been proposed to detect and track real-world events from traditional media outlets. The majority of these techniques try to detect and extract event information by generating potential event clusters, where each cluster contains objects describing the same event. We refer to these objects as *event entities*. Two types of event entities are recognized in the literature, namely, documents and features. Therefore, event detection approaches can be classified into two major categories based on the event entity type:

(1) **Document-pivot**: the events are detected by clustering documents on the basis of some established similarity criteria [35, 55, 124, 151, 156].

(2) **Feature-pivot**: representative features, e.g., bursty words, are extracted, and then clustered to form potential events [46, 47, 53, 75, 90].

2.5.1 Document-pivot Detection

The core of this line of event detection approaches is to assign documents about the same event to the same cluster. For this, a sequence of standard steps is usually followed [162], namely, content preprocessing, document representation, and document organization (clustering), as illustrated in Figure 2.4. Data preprocessing includes sentence tokenization, stop-word elimination, and word stemming. For data representation, the Vector Space Model (VSM) is basically used, representing a document as a vector of terms (*term vector*), each of which is given a weight. A term is weighted based on the traditional Term Frequency-Inverse Document Frequency

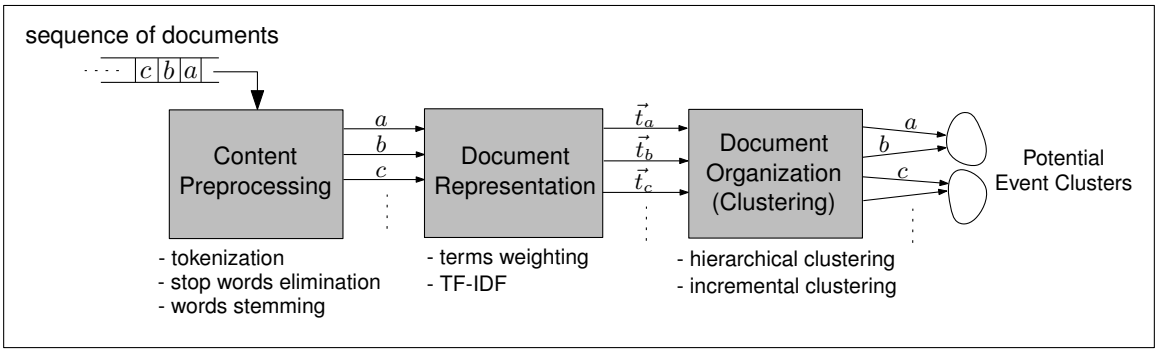


Figure 2.4: Main steps of document-pivot detection frameworks.

(TF-IDF) weighting scheme [133], which combines both a local term weight (TF) with a global one (IDF). Several variations of TF-IDF have been proposed, where according to [161], the *ltc* alternative yielded the best detection results. The weight of term t in document d based on the *ltc* weighting scheme is defined as

$$w_{t,d} := \frac{(1 + \log_2 tf_{t,d}) \times \log_2(\frac{|D|}{n_t})}{\|\vec{d}\|} \tag{2.2}$$

where $tf_{t,d}$ is the frequency of term t in document d, n_t is the number of documents containing term t, $|D|$ is the number of documents in the corpus D, and $\|\vec{d}\|$ is the 2-norm of vector $\vec{d}$, which acts as a normalization factor and is defined as

$$\|\vec{d}\| := \sqrt{\sum_t w_{t,d}^2}. \tag{2.3}$$

The factor $(\log_2(\frac{|D|}{n_t}))$ in Eq. 2.2 is the inverse document frequency (IDF) whose value increases as the number of documents containing term t decreases. The rational behind using the global weight (IDF) is that the less documents a term is observed in, the more significant the term is. In the last step, document organization (clustering), and after normalizing the weights in term vectors, the similarity between these vectors are quantified based on some similarity measures (e.g., cosine similarity and Pearson correlation) and used to organize documents into clusters, each of which refers to a potential event.

Estimating the similarity between documents is a key operation for text clustering [9]. The similarity between two documents (a, b) corresponds to the correlation between their respective term vectors $(\vec{t_a}, \vec{t_b})$, and is quantified as the cosine of the

angle between these vectors [82], i.e.,

$$cosine(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a} \cdot \vec{t_b}}{\|\vec{t_a}\|\|\vec{t_b}\|}, \tag{2.4}$$

where $\|.\|$ is the 2-norm of the enclosed vector. Since the weights in a term vector are non-negative, the computed cosine similarity is also non-negative and bounded between $[0, 1]$. A key characteristic of cosine similarity is its independence of document length, making it one of the most popular document similarity measure, especially in the field of information retrieval [23] and document clustering [97].

Unfortunately, the VSM model ignores the hidden semantics (e.g., named entities and synonyms), the syntactic features (e.g., parts of speech), and the order of words in each document. Moreover, the large dimensionality of term vectors adversely affects a proper estimation of the similarity (dissimilarity) between documents, in particular, when long text documents are used. To reduce the dimensionality of term vectors and choose the most informative terms, Latent Semantic Analysis (LSA) [111] is employed, leading to a better performance in clustering topically-related documents. Nevertheless, LSA cannot distinguish between events related to the same topic, namely, when a large number of words are common between these events, e.g., the occurrence of two different car accidents. To remedy this, both temporal and textual information are jointly used in document clustering [160, 162] for both the retrospective (RED) and online (NED) event detection.

Yang et al. [162] adopt an agglomerative-based group-average clustering (GAC) for RED. First, temporally-close documents are placed into one bucket, giving a higher priority to grouping consecutive documents. The clustering algorithm works in a bottom-up fashion, whereby at each iteration, the process repeats and generates clusters at higher and higher levels until some predefined top-level clusters are obtained. On the other hand, for NED, they use a time window to limit the number of documents under investigation. An incremental clustering algorithm is used to organize document into clusters. A new document x is absorbed by the most similar cluster if the *confidence score* between x and the seed of that cluster is below a predefined threshold; otherwise, the document x is treated as a seed for a new cluster. The confidence score of document x is computed as follows:

$$score(x) := 1 - \arg\max_{i \in window} \left\{ \frac{i}{m} \times cosine(\vec{t_x}, \vec{t_i}) \right\} \tag{2.5}$$

where the fraction $\frac{i}{m}$ is treated as a decaying weight such that old documents have progressively less impact on the current decision. Similarly, the approach by Allan et al. [14] processes a stream of documents in a strict online settings to detect new events and track existing ones using a single-pass clustering algorithm. In addition to incorporating temporal information besides content, a novel event-level thresholding model is introduced to obtain adaptive confidence scores for each document when compared with the seeds of existing clusters. A 2-stage approach for a better NED has been proposed by Yang et al. [163]. First, each arriving document is classified into a broad topic category. Then, named entities are extracted from each document, acting as event-level features that are helpful in distinguishing between on-topic documents but relating to different events. The weights of the named entities are optimized relative to normal words for each topic.

A key issue for obtaining high-quality event clusters is the reliability of the adopted term weighting scheme. Hence, several refinement steps have been introduced, trying to come up with more informative and semantic-aware weighting schemes. Brants et al. [37] found that documents from different sources have somewhat different vocabulary usage. Therefore, a source-specific TF-IDF model has been applied where each source has its own TF-IDF model. Furthermore, they have adjusted the way of measuring the similarity between the current document x and document i by including the average similarity $E_{s(x),s(i)}$ of documents on the same event, taking into account the source pair $<s(x), s(i)>$ that x and i are drawn from, respectively.

Kumaran and Allan [93] have shown that measuring the degree of overlap between documents using cosine similarity might lead to erroneous results when the documents are represented using the traditional TF-IDF-based weights. For more reliable results, two additional term vectors are created for each document; one for named entities and the other for non-named entities. For further improvements in weighting terms, He et al. [77] make use of bursty features along with their bursty periods, which are extracted from the document stream. First, bursty features are identified using the method of Kleinberg [90] and then the TF-IDF weights are updated by boosting those pertaining to bursty features by a certain factor.

Recall that tracking an event requires finding unlabeled documents (stories) that can be assigned to the cluster of this event based on a certain similarity measure (see Section 2.3.2). Apart from tracking events using unsupervised approaches, event tracking can be reduced to a text classification problem [160, 161]. A major difficulty in this track is the lack of positive training instances (documents) per event since most of the documents are unlabeled. What exaggerates the problem is that the majority of

events have a short time duration. To handle this, Yang et al. [160] introduced several new variants of the Ricchio approach [131] and the k-Nearest Neighbor (kNN) [73] algorithm in order to prevent negative examples from dominating the results of the used scoring functions. As a result, their overall event tracking performance improved and became one of the top-performing methods in the TDT3 official evaluation.

Li et al. [106] proposed a probabilistic model for the RED task, where an Expectation Maximization (EM) algorithm is adopted to learn the model parameters and to maximize the log-likelihood of the distribution function. However, besides being computationally infeasible when applied to a large corpus, such algorithm requires the number of events to be given a priori.

2.5.2 Feature-pivot Detection

An event detection framework based on feature-pivot detection methodology consists of three main steps:

(1) **Feature extraction**. The content of each document is preprocessed and then features (e.g., unigrams, bigrams, and n-grams) along with their respective count statics are extracted. Several methods have been used to extract features from documents, ranging from a simple single-pass ones, to a more complex optimization-based extraction [104].

(2) **Feature selection**. Among the large number of extracted features, those that can be identified as event entities are selected. Bursty features that show a sudden surge in usage over a period of time are typical examples of features selected as candidate event entities. For this, a variety of bursty feature selection techniques are adopted, e.g., techniques based on the discrepancy principle [4, 96], wavelet analysis [156], and Discrete Fourier Transform [76].

(3) **Feature clustering**. In this step, the selected bursty features are clustered using, e.g., model-based [171], graph-based [108, 164, 168], or density-based [35] clustering algorithms.

Document-level TF-IDF-based weighting schemes assign relatively high weights to terms that have multiple occurrences in a small number of documents. However, when the popularity of a certain topic starts increasing (becoming a trend), a large number of documents containing related terms appear, discussing related events and activities. In this case, TF-IDF performs poorly as key terms will get low weights due to the impact of the global factor (IDF). What makes the problem more severe

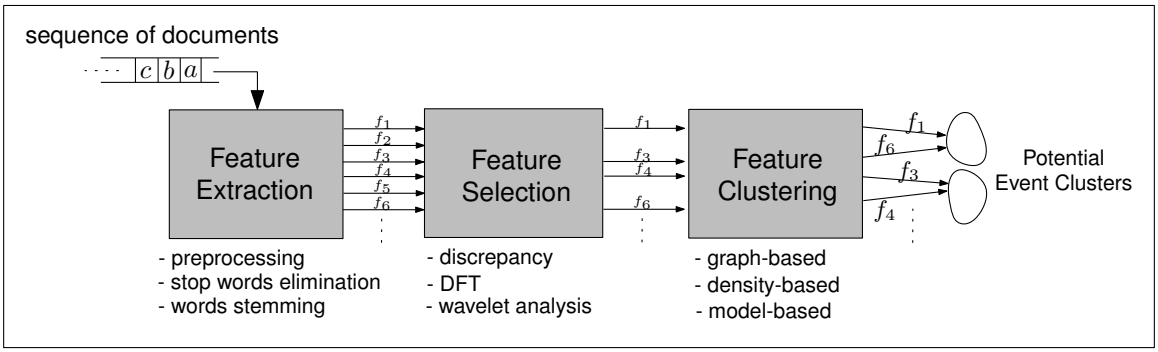


Figure 2.5: Main steps of feature-pivot detection frameworks.

is when streaming algorithms are adopted, in particular, using a sliding window over the most recent documents. To handle this, Bun and Ishizuka [40] leveraged term occurrences in multiple channels and proposed a different term weighting scheme called TF-Proportional Document Frequency (TF-PDF). It assigns higher weights to terms appearing more frequently in many documents on several channels and lower weights to the terms that are rarely mentioned. In spite of this added value of the TF-PDF measure, it is unable to capture the variations in term popularity over time. Therefore, Chen et al. [45] combined the TF-PDF measure with measures from the Aging Theory [44] in order to build their own bursty word extraction framework with more reliable results.

The core of feature-pivot detection is to find bursty features from a stream of documents, and then, to group these features based on some common characteristics, e.g., grouping features with similar temporal patterns [62, 90]. Kleinberg [90] proposed an infinite-state automaton to model the inter-arrival times of a stream of documents in order to extract bursty words showing significant increase in usage. Fung et al. [62] model the temporal distribution of words in a corpus as a binomial distribution. Based on this model, bursty words are identified by a heuristic-based thresholding and then strongly interrelated bursty features are grouped, forming potential events.

He et al. [76] built a model to classify each word (called feature) along two categorical dimensions: periodicity and power, after converting its signal from the time domain into the frequency domain using Discrete Fourier Transformation (DFT) [143]. Their approach identifies the bursty period among the entire trajectory of each feature having a high spectrum power from one side or long-term periodic or aperiodic nature on the other side. For this purpose, a feature's trajectory is modeled with a Gaussian distribution, and the bursty period is chosen to be the one whose signals fall

within one standard deviation from the mean. However, this spectral-based approach does not scale well for a large number of streaming documents.

Lee et al. [101] built a new term weighting scheme, assessing the burstiness degree of a term in terms of skewness, consistency, periodicity, and variation. By combing these factors into one measure, they showed that their approach outperforms the solutions by [45, 62, 76, 157] with respect to the quality of the results.

2.6 Event Detection from Social Media

As discussed in Section 2.2.1, user-generated content published via several social media services is a huge source of continuous and noisy textual messages. Although this content hides useful event information, extracting it is a challenging task. Event detection from social media is currently a hot research area, which draws on techniques from various disciplines such as data mining, natural language processing, information retrieval and extraction, machine learning, spatio-temporal analysis, to name but a few. One can find related publications in conferences and journals with diverse research directions, and consequently, there are a large number of research contributions on event detection from social media. In this section, however, we do not provide an exhaustive survey of existing techniques but rather select and discuss representative ones. For each approach, we study (1) the methodology employed to detect events and (2) the main objectives of the detection process. Based on these aspects, the wide range of event detection approaches are classified according to the *detection methodology* (Section 2.6.1) and the *detection objectives* (Section 2.6.2). Note that these categories and subcategories that we will cover later are not mutually exclusive, but rather represent either the type of the detection approach or the type of the detected event.

2.6.1 Categorization By Methodology

We classify the major methodologies proposed for event detection along three categorical dimensions (see Figure 2.6): (I) Event entity type (document-pivot versus feature-pivot techniques); (II) Detection task (NED versus RED techniques); and (III) learning methodology (supervised versus unsupervised techniques).

Table 2.1: Comparison of several event detection approaches based on the used detection methodology.

REFERENCES	DETECTION TASK		EVENT ENTITY TYPE		LEARNING MODEL	
	NED	RED	Feature-pivot	Document-pivot	Supervised	Unsupervised
Parker et al. [122]		✓	✓		✓	
Guille and Favre [71]	✓		✓			✓
Valk. and Gunop. [151]	✓			✓	✓	
Petrovic et al. [124]	✓			✓		✓
Weng and Lee [156]		✓	✓			✓
Li et al. [105]	✓			✓	✓	
Zhang et al. [168]		✓	✓			✓
Albakour et al. [10]	✓			✓		✓
Sayyadi [136]		✓	✓			✓
Parikh and Karlapalem [121]	✓		✓			✓
Qin et al. [125]		✓	✓		✓	
Lee [98]	✓		✓			✓
Boettcher and Lee [35]	✓		✓			✓
Lee and Sumiya [100]	✓			✓		✓
Alvanaki et al. [15]	✓		✓			✓
Mathioudakis et al. [113]		✓	✓			✓
Cataldi et al. [41]		✓	✓		✓	✓
Mathioudakis and Koudas [114]	✓		✓			✓
Walther and Kaisser[152]	✓	✓		✓	✓	✓
Aggarwal and Subbian [8]	✓			✓	✓	✓

Table 2.2: Comparison of several event detection approaches based on the used detection methodology (continued).

REFERENCES	DETECTION TASK		EVENT ENTITY TYPE		LEARNING MODEL	
	NED	RED	Feature-pivot	Document-pivot	Supervised	Unsupervised
Sakaki et al. [130]	✓			✓	✓	
Li et al. [103]		✓	✓			✓
Becker et al. [28]	✓			✓	✓	✓
Sankaranarayanan et al. [135]	✓			✓	✓	✓
Xu et al. [158]		✓	✓		✓	
Lappas et al. [96]	✓		✓			✓
Chen and Roy [46]		✓	✓			✓
Lee et al. [99]	✓		✓			✓
Shan et al. [139]	✓			✓		✓
Kwan et al. [94]		✓	✓			✓
Skovsgaard et al. [142]	✓		✓			✓
Watanabe et al. [154]	✓		✓			✓
Budak et al. [39]	✓		✓			✓

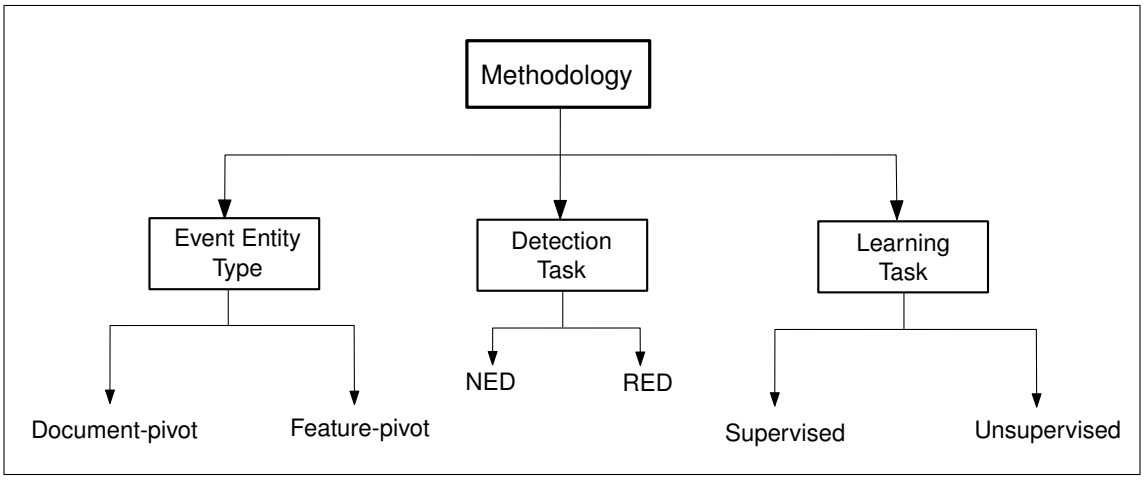


Figure 2.6: Categorization by methodology for event detection approaches.

Event Entity Type

Document-pivot techniques. In Section 2.5, we showed that document-pivot techniques have been widely employed to detect events from text streams by clustering event-related documents. In the context of social media, however, this line of techniques has the following limitations:

(1) Microblogs are inherently brief updates, which implies that although when some microblogs pertain to the same event, they might share no keywords [45]. As a result, and based on the typical TF-IDF weighting scheme, the distance between their corresponding term vectors will be large. Hence, microblogs on the same event might be assigned to different clusters. On the other hand, the chance of having off-topic microblogs assigned to the same cluster is relatively high, in particular when some ongoing events share a number of keywords, e.g., when two soccer matches take place at the same time.

(2) TF-IDF weighting scheme is originally designed for tasks in Information Retrieval [23, 111] even though it performs well in text clustering problems [62]. Both the local (TF) and global (IDF) parts might result in unreasonable weights in the context of social media. For the TF part, an event-related keyword will generally obtain a small value in a microblog, because it is unusual to find a microblog having multiple occurrences of this keyword. On the other hand, and because of the noisy nature of social media, the existence of a very few occurrences of a certain word does not indicate that this word is significant, which contradicts the role of the global IDF part.

(3) The TDT research direction assumes that all documents are relevant to some old or new events of interest [13]. However, in social media, microblogs on real-world events are buried in a huge amount of noisy data (see Section 2.2.3).

Therefore, the majority of approaches on event detection from social media fall under the feature-pivot category. However, some research efforts have adapted a number of document-pivot techniques to cope with the uniqueness of social media content. Shan et al. [139] proposed a novel burst-based VSM for a better representation of documents. This is because the traditional VSM does not account for the temporal dimension, which is essential in distinguishing between on-topic events as discussed in Section 2.5. Bursty features are extracted from the stream using the approach by Kleinberg [90], where only these bursty features are used to build the term vector of each document in the stream.

Petrovic et al. [124] introduced a system on the first story detection task (FSD) from a stream of Twitter posts using *locality-sensitive hashing* (LSH) [126]. When a new tweet arrives, it is hashed into a bucket and the similarity between this tweet and the ones in that bucket is calculated using cosine similarity. If the similarity to the closest tweet is below a certain threshold, the tweet is placed in a separate bucket and treated as a first story of a potential event; otherwise, the tweet is placed in the bucket of the most similar tweet. Due to the noisy content and the short length of tweets, a variation-handling procedure is adopted to improve the overall system accuracy. Fast-growing clusters are then reported as real-world events.

Valkanas and Gunopulos [151] employ sentiment analysis based on notions from emotional theories to detect events from Twitter streams. The rationale is that messages published about an event usually convey the emotional state of the users affected by this event. First, each incoming tweet is classified into one of seven emotional classes (anger, fear, disgust, happiness, sadness, surprise, and none). When a surge in any emotional state is observed at a certain location, an event is reported at that location and some term-weighting techniques, e.g., [42] and [27], are used to extract representative keywords for each event.

In order to avoid considering the entire stream of messages when a specific event type is investigated, Sakaki et al. [130] built a supervised classifier to filter out the tweets that are irrelevant to a target event. Event-related tweets are then modeled over time using a Poisson process to detect the occurrence of an event. They adopt a temporal model that depicts the probability of a target event occurs at time t

according to the following formula:

$$\mathrm{p}_{occur}(t) = 1 - \mathrm{p}_f^{g(t)}, \tag{2.6}$$

where

$$g(t) = \frac{n_0(1 - \exp(-\lambda(t+1)))}{(1 - \exp(-\lambda))} \tag{2.7}$$

and p_f refers to the probability that the expected number of sensors $g(t)$ will generate a false prediction at time t. The number of sensors producing predictions at earthquake time is exponentially distributed, and n_0 is the number of sensors at time 0. Using their approach, when a new type of events is considered, the classifier needs to be trained using a different set of event keywords, which is a limitation that stands against applying their approach to generic events.

Feature-pivot Approaches. The basic idea of feature-pivot detection (as discussed in Section 2.5.2) is to extract features (e.g., words, bigrams, and phrase segments) from microblogs, to select a subset of these features as event candidates, and then to group these features into clusters of potential events. Due to the high uncertainty and noisiness of social media content [158, 165], the expected number of resultant clusters is large, where most of these clusters are non-events (low precision). This undesirable clustering of event entities is due to the fact that people share various types of content such as personal updates, opinions, and random thoughts [118]. Moreover, assuming that people share some event-related messages, the uncertainty in what they publish, when, and where is very high. For example, they might publish messages on a certain event after it ends. Several solutions have been proposed in an effort to mitigate this problem and to improve the overall precision. These solutions involve one or a combination of the following directions:

(1) *improving feature selection.* The temporal distributions of features in social media are very noisy; and hence, not all selected features refer to some events [151]. The aim is thus not to miss the identification of any event-related feature and to minimize the number of "spurious" features, which results in high recall and precision, respectively. A wide variety of feature selection techniques have been proposed for this sake. Some of them are based on a simple, yet efficient threshold-based bursty feature extraction [3, 96, 157]. Others study the burstiness of bigrams [15, 121], or even phrase segments [103]. In addition, more

sophisticated signal processing techniques are used to detect spikes in feature usage over time, e.g., using Wavelet-based analysis [156].

(2) *using noise-resilient clustering.* This can be achieved by choosing a clustering algorithm that is robust against outliers, e.g., DBSCAN [73].

(3) *supervised filtering of clusters.* The generated clusters are classified into events or non-events using a classifier trained on an annotated set of features [35, 125, 152].

In [156], a wavelet-based signal processing framework, called *EDCoW*, is proposed to capture important differences in the "energy" of individual words in a sliding window. To perform word clustering and generate potential events, the cross correlation between signals are measured, resulting in a symmetric correlation matrix. From graph theoretical point of view, this matrix can be viewed as the adjacency matrix of an undirected graph $G = (V, E, W)$. A modularity-based graph partitioning is exploited to cut the graph G into subgraphs and generate event clusters. Assume that the cross correlation between the words i and j is denoted X_{ij}. The modularity of the partitioning is defined as

$$P = \frac{1}{2m} \sum_i \sum_j (X_{ij} - \frac{d_i \times d_j}{2m}) \delta_{c_i,cj}, \tag{2.8}$$

where $d_j = \sum_i X_{ij}$, $m = \sum_i \frac{d_i}{2}$ is the sum of all the edge weights in G, $c_{i(j)}$ is the index of the subgraph that node $v_i(j)$ belongs to, and $\delta_{c_i,cj}$ is the Kronecker delta such that $\delta_{c_i,cj} = 1$ if $c_i = cj$ and $\delta_{c_i,cj} = 1$ otherwise. The goal now is to split the graph into subgraphs by maximizing P, which is solved using the approach proposed by Newman [119]. This approach is based on reformulating the modularity in terms of the eigenvectors of a new modularity matrix, resulting in more reliable results in shorter running time. A major drawback of *EDCoW* is that building a signal for each word and computing the cross correlation for each pair is computationally expensive. To cope with this, the majority of words are filtered out based on a statistical thresholding criterion. Unfortunately, this might lead to ignoring important words belonging to events with sparse observations, i.e., small-scale events.

Likewise, Cataldi et al. [41] built a model in support of connecting emerging keywords after navigating a topic graph. For this, they first assign a weight for each word in a microblog using the augmented normalized term frequency [132]. To account for the credibility of the source, namely, the publishing user, the social interactions

are analyzed with the well-know Page Rank algorithm [38]. After that, a novel aging theory is used to mine emerging keywords. From these keywords, a set of emerging topics are finally identified based on the keyword co-occurrences inferred from a topic graph.

Li et al. [103] introduced *Twevent*, a framework to detect real-world events from tweets. First, segments (phrases that are semantically meaningful combinations of words) are extracted from tweets. Then, bursty segments are identified within a time window based on their frequencies, and finally, these bursty segments are clustered using Jarvis-Patrick algorithm [84]. They apply a heuristic-based filtering to distinguish between event from non-event clusters using an external information source, namely, Wikipedia. This type of filtering, however, leads to ignoring events that are not mentioned in the used external source. Instead of using heuristic-based techniques for filtering out non-event clusters, Qin et al. [125] adopt a supervised approach, classifying the generated clusters into events or non-events. The selected features fall under three categories: statistical, social, and textual. These features are assumed to be indicative of the significance of examined event clusters. The 10-fold cross validation is utilized in the model learning process, which led to a precision of 0.84 compared to 0.76 by *Twevent* [103].

Parikh and Karlapalem [121] presented *ET*, another event detection framework following a detection paradigm similar to *Twevent* [103]. However, to improve the computational efficiency, *ET* (1) confines the extracted segments to bigrams, avoiding a computationally expensive optimization; (2) uses an index, maintaining the bursty time blocks of each segment; (3) exploits an agglomerative hierarchical clustering to group the segments that are related to the same event. This clustering is applied to a segment similarity matrix that encodes the similarity between each pair of segments. The similarity of each pair of segments is quantified based on both their content and temporal patterns. This type of clustering in both content and time has proven to be efficient in detecting events. For example, in [135], a variation of the leader-follower clustering algorithm was adapted to work in an online fashion and to account for the temporal order of incoming microblogs.

Mathioudakis and Koudas [114] introduced TwitterMonitor, a two-step approach to extract trends from a Twitter stream and, in a third step, to analyze the detected trends. In the first step, bursty words are identified based on the queuing theory. Then, these bursty words, using their co-occurrences in the most recent tweets, are clustered to form potential trends. This bursty word clustering task utilizes a greedy strategy in order to generate groups of bursty words using a small number of op-

erations, which alleviates the computational overhead of tweets having high arrival rates. Instead of treating unigrams as features to be clustered, Alvanaki et al. [15] exploit tag-pairs associated with tweets to identify emergent topics. A sliding window on a stream of microblogs is used to compute statistics about tag-pairs, which are then used to identify unusual shifts in correlations between such tags, based on the fact that these shifts are mainly driven by real-world events. To remedy the computational overhead caused by the pair-wise correlation computation, a subset of the tags having high popularity are selected and used as seed-tags. For further tagging enrichment, they make use of an entity tagging approach that extracts named entities (people, organizations, and places) by identifying the phrases for which an article in Wikipedia exists.

Detection Task

In Section 2.3, we discussed both NED and RED detection techniques, stating that while RED techniques retrospectively extract events from a corpus of documents, NED is tailored to process documents continuously as they arrive using streaming algorithms. In contrast to RED, it is apparent that the NED approaches allow for an immediate detection of real-world events, which is critical for different application scenarios, e.g., detecting the occurrence of earthquakes in real-time [130].

NED. Using NED techniques, event entities are processed incrementally to either report the occurrence of a new event or track existing ones (see Section 2.3). A major challenge for applying streaming algorithms to social media content is the high-arrival rate of microblogs, which not only results in too many non-event clusters, but also requires huge computational resources. To remedy this, a number of techniques or a combination of them have been proposed. Some of these techniques are described as follows:

(1) *Adapted feature selection*: Apart from traditional bursty feature selection methodologies that are based on identifying features with sudden surge in usage (see Section 2.5.2), other types of selection mechanisms have been proposed: (I) Heuristic-based techniques: They involve selecting features that are assumed to be thematically representative of their respective events. For example, Watanabe et al. [154] assume that localized events can be detected by extracting place names (georeferences) and then forming clusters that contain these place names along with co-occurring keywords. Moreover, source-specific

keywords are selected and utilized as potential features, i.e., event entities. For instance, in [87] and [108], hashtags are selected as event entities, describing ongoing events. Guille and Favre [71], on the other hand, focused only on dynamic links (mentions) observed in tweets. (II) supervised techniques (see "Supervised Detection" in Section 2.6.1). Here, we argue that selecting specific types of features leads to ignoring a wide range of keywords that belong to small-scale event. In addition, the semantics of detected events will be degraded when only focusing on a certain type of features.

(2) *Locality-sensitive hashing (LSH)*: It is a randomized technique for finding the nearest neighbor in vector space of a query point (e.g., microblog) in sub-linear time [126]. For example, Petrovic et al. [124] adopt LSH to efficiently determine if an arriving tweet represents the onset of a new event (FSD) or can be assigned to an existing bucket (cluster).

(3) *Sliding Windows*: In many application scenarios, the most recent data is considered to be most important. Therefore, the majority of NED approaches adopt a sliding window on a stream of microblogs, (see, e.g., [2], [99], and [110]). When the window moves, outdated (expired) data is eliminated and new observations are considered for further analysis.

(4) *Randomized data structures*: It is a class of data structures where each acts as a frequency table of events[1], absorbing a stream of data items. Hash functions are used to map events to frequencies. A key characteristic of this class of structures is that only sub-linear space is required while guaranteeing accuracy. Sketch tables [6] are the most prominent randomized data structures that have been used in the context of social media analysis to efficiently digest the large number of incoming microblogs, see e.g., [39] and [8]. In [8], the authors used a sketch table to efficiently compute the structural similarity between clusters.

(5) *Sampling*: To avoid digesting the entire sequence of arriving microblogs, only a representative sample within a sliding window is maintained. An example of approaches adopting this technique is described in [151], where the "chain-sample" algorithm [20] is used.

(6) *Indexing*: Providing proper index structures for managing microblogs arriving with high rates is important towards building practical detection frameworks [2,

[1] event entities in this context

109, 135]. For example, Sankaranarayanan [135] chose to reduce the number of distance computation operations required for online clustering using an inverted index that confines the search to only those clusters that contain the features (keywords) of the microblog to be clustered [135]. As for further optimization, a list of active clusters, whose centroids are less than three days old, is maintained as candidates to which a new tweet can be added. Magdy et al. [110] use a hierarchical space-partitioning index structure (Pyramid [147]) that supports the insertion of microblogs with high arrival rates and is able to expel outdated microblogs efficiently.

(7) *Parallel Processing*: Nguyen and Jung [120] employed the well-known Apache Hadoop[1] framework to distribute the computation and storage required to process microblogs arriving during a time window among multiple processing nodes.

It is noteworthy that most NED techniques receive a chronologically-ordered corpus of microblogs to detect and analyze past events.

RED. Although most research discussed so far focuses on NED techniques for event detection, making use of the timestamps associated with microblogs, some interest has recently arisen in extracting events retrospectively from historical data. Hong et al. introduced a generative model to reflect the generation of tweets across geographic space with applications on content recommendation and topic tracking [80]. Zhang et al. [166] went beyond feature co-occurrence. They extract spatial, temporal, and spatio-temporal distributions of tags, and then represent them as vectors that can be clustered to help users recognize the semantic relationship of tags.

Xu et al. [158] deal with recovering scarce and non-uniformly distributed phenomena over spatio-temporal bins from social media. They reduce the problem of signal recovery into a Poisson point process estimation. However, their approach is devoted to static datasets and will perform poorly when applied to fine-grained spatio-temporal settings.

Chen and Roy [46] exploit the spatio-temporal information of photos in Flickr to discover tags related to events. They use Discrete Wavelet Transform to emphasize *dense spatial regions* and suppress noise in the 3D space (location and time). Their approach, however, operates at coarse spatio-temporal granularity (1-day time intervals and 1-degree spatial shifts).

[1] `http://hadoop.apache.org/`

Learning Methodology

The task of event detection is typically performed using unsupervised clustering-based approaches [18]. However, supervised learning is sometimes used prior event detection to eliminate irrelevant entities and/or to filter out non-event clusters in an effort to mitigate the noisy nature of social media.

Unsupervised detection. The majority of techniques for unplanned event detection rely on clustering approaches, as shown in Table 2.6. Manually labeled data is not required for training purposes, and thus unsupervised techniques are more suitable to the problem of event detection, which is inherently a query-free problem. The clustering mechanism should be efficient and highly scalable given the increasingly large amounts of content published via social media sites. The approach by Abhik and Toshniwal [5] first selects event-related messages from different social media sources using a simple rule-based mechanism. Then, the selected messages are clustered using a single-pass threshold-based algorithm, see, e.g., [28], [124], and [135]. Incremental clustering is more suitable for grouping continuously generated event entities. This is performed by setting a maximum similarity between new entities and any of the existing clusters. Aggarwal and Subbian [8] leverage both the underlying network structure of social media and the content of messages to provide a more effective clustering scheme and to improve upon the content-only similarity metric.

Gao et al. [65] studied the spatio-temporal distribution of microblogs and analyzed their content in Sina Weibo[1] Website on realistic datasets. An adaptive K-means clustering algorithm is used to group tweets published about a real-world event at a certain geographic area. In [10], a novel event retrieval framework is devised to reliably identify and rank real-world events. For this purpose, the framework combines (1) evidence from the content of microblogs, namely, how relevant the tweets that are posted from a certain location at a specific time to a user query and (2) the change point analysis based on Grubb's test [70]. The system described in [155] detects the occurrence of events at a certain location by monitoring the frequencies of terms in tweets across time and space. The log-likelihood ratio test is employed to quantify the change in the usage behavior of terms. The extracted terms are then combined based on the observed co-occurrence patterns to form potential events.

[1] `http://weibo.com/`

Unsupervised event detection has also been considered through graph analysis. Zhao and Mitra [170] define a real-world event as a set of relations between users on a specific topic during a certain time interval. They represent the social text streams as a graph where each node corresponds to a social actor and each edge refers to the similarity between the textual content of the corresponding actors. Events are extracted by combining text-based clustering, temporal segmentation, and graph cuts of social networks. Sayyadi et al. [136] model the co-occurrence relationship between extracted keywords using a graph, where each node reflects a keyword and an edge between two nodes indicates that the corresponding keywords co-occur in a number of documents above some threshold. By analogy, the graph is thought of as a social network of relationships between keywords, whereby a community detection technique based on the betweenness centrality [165] is used to extract connected components from the graph. Each such component represents a keyword summary of a potential event.

Supervised Detection. As stated earlier, in the context of event detection, supervised techniques are mainly employed in one of the following:

(1) Filtering out spurious features (or documents) so that only event-related ones undergo clustering. Although manually annotating a large number of microblogs is a labor-intensive and time-consuming task, it is more feasible for detecting specified events where some event descriptions, such as keywords, location, or time are known in advance. As a result, this would reduce the number of microblogs messages that must be processed and allow the detection framework to focus on a restricted set of microblogs. Li et al. [105] built a classification model to increase the accuracy of extracting CDE-related[1] tweets. They found that the performance of their system is not acceptable when only the textual content of tweets is considered as features for the classifier. Hence, they additionally exploit Twitter-specific tokens, e.g., hashtags, and CDE-specific features like whether a tweet contains locations, numbers etc. The model by Sakaki et al. [130] annotates each tweet as relevant or irrelevant to an event using SVM where three set of features for each tweet are used: statistical, keyword, and word context features.

(2) Classifying already generated clusters into event or non-event clusters. In [152], a classifier is trained based on 41 features, including textual features, user

[1]CDE: Crime- and Disaster-related events

counts, tweet counts, and other spatial-centric features. In addition to the traditional temporal and topical features, the classification model proposed by Becker et al. [28] incorporates social features, such as the percentage of tweets containing retweets, replies, and mentions in a cluster for a better classification capability.

2.6.2 Categorization By Objectives

Real-world events of different types, levels of detail, and associated meta-data (e.g., start time or location) are extracted from the content of social media. Some of the approaches, for example, focus on detecting planned events and others on specifying the location of the detected events. In general, the heterogeneous nature of both the detected events and associated metadata plays a big role in designating the type of detection approaches to use. Therefore, the systems on event detection can be further categorized into the following: (I) query-driven versus query-free, referring to whether a detected event meets a query posed by the user (query-driven), or the system can detect the occurrence of any type of event (query-free); (II) awareness of event location; and (III) planned versus unplanned event detection.

In this section, we address some representative approaches on event detection, which are query-driven, location-aware, or devoted to extracting planned event, since the other detection dimensions (query-free, location-unaware, unplanned event detection) are more popular and covered in different sections of this chapter. See Table 2.3 for a list of efforts classified based on the aforementioned objectives.

Query-driven Detection

Usually, no information about the events to be detected is given a priori, and hence, event detection is basically defined as a discovery problem (see Section 2.3), where query-free-based approaches are typically employed as a solution for such a problem. Nevertheless, some efforts try to exploit query-driven approaches in order to detect and track some specified events, e.g., detecting the occurrence of car accidents. Query-driven techniques prune the huge space of messages and process only those microblogs that contain some predetermined keywords, which helps filter out noisy and irrelevant microblogs at an early stage. However, the main shortcoming of query-driven techniques is that a different query has to be provided each time a new type of events is considered. Query-driven detection is therefore somewhat more linked to Information Retrieval than to pattern discovery.

Table 2.3: Comparison of several event detection approaches based on the objective of the study.

REFERENCES	QUERY-		LOCATION-AWARE	
	driven	free	geo-coordinates	place names
Parker et al. [122]	✓			
Guille and Favre [71]		✓		
Valk. and Gunop. [151]		✓		✓
Petrovic et al. [124]		✓		
Weng and Lee [156]		✓		
Li et al. [105]	✓		✓	✓
Zhang et al. [168]		✓		
Albakour et al. [10]	✓			✓
Sayyadi [136]		✓		
Parikh and Karlapalem [121]		✓		
Qin et al. [125]		✓		
Lee [98]		✓	✓	
Boettcher and Lee [35]		✓	✓	
Lee and Sumiya [100]		✓	✓	
Alvanaki et al. [15]		✓		
Mathioudakis et al. [113]		✓	✓	
Cataldi et al. [41]		✓		
Mathioudakis and Koudas [114]		✓		
Walther and Kaisser [152]			✓	
Aggarwal and Subbian [8]		✓		
Sakaki et al. [130]	✓		✓	
Li et al. [103]		✓		
Becker et al. [28]		✓		
Sankaranarayanan et al. [135]	✓			✓
Xu et al. [158]	✓		✓	
Lappas et al. [96]		✓	✓	
Chen and Roy [46]		✓	✓	
Lee et al. [99]		✓		
Shan et al. [139]	✓			✓
Kwan et al. [94]		✓		
Skovsgaard et al. [142]	✓		✓	
Watanabe et al. [154]		✓	✓	✓
Budak et al. [39]		✓	✓	

A news processing system, called TwitterStand [135], was built to extract and describe news from Twitter streams. In order to involve only news-related tweets in the clustering process and to alleviate the undesirable consequences of noise, a naive Bayes classifier [32] is used to discard tweets that clearly cannot be news. Parker et al. [122] tried to track public health trends from medical tweets, where Wikipedia is utilized to associate trending word sets with medical topics and filter out non-medical trends. Then, medical trends are aggregated to detect shifts in public health conditions.

TEDAS [105] is a framework to detect CDE (crime- and disaster- related events). First, tweets are filtered to identify related tweets using a rule-based information retrieval architecture that is based on a novel query expansion. The identified tweets are then clustered and the clusters are ranked. Due to the short length of microblogs, query expansion is useful for retrieving microblogs relevant to the initial query posed by a user. For example, Chen et al. [48] present a novel algorithm to generate and rank candidate expansion terms that are relevant to a topic word posed as a query to the event detection system.

Marcus et al. [112] built *TwitInfo*, a Web portal to visualize and summarize tweets using a timeline-based tweet counting. When a user posts a keyword, the system draws a diagram showing the tweet frequency over time. Then, an off-line peak detection algorithm is adopted to highlight hot spots that correspond to potential real-world events.

Another line of query-driven approaches use instead spatio-temporal meta-data as queries specifying the spatio-temporal context within which events might occur. Skovsgaard et al. [142] introduce a scalable framework with the purpose of processing top-k most popular term queries on a stream of microblogs in a user-specified spatio-temporal range. A variation of the count-based frequent item algorithm, SpaceSaving [116], is proposed to adaptively maintain exact counts for terms at various spatio-temporal granularities. Magdy et al. [110] present *Mercury*, a system to support real-time top-k spatio-temporal queries on microblogs within a memory-constrained environment. *Mercury* helps retrieve microblogs related to a certain event provided that the event location and time are known a priori.

Location-aware Detection

An event is basically defined as "something that happens at some specific time and place" [161]. The process of event detection involves therefore the identification of what the detected events are about, when they started/ended, and where they took

place, using the content of related messages along with the associated spatio-temporal metadata. To date, most of the research efforts that concentrate on detecting events consider only the textual content of messages and the attached timestamps. However, the proliferation of global positioning systems and the prevalence of GPS-equipped mobile devices have recently increased the percentage of geo-tagged messages. For example, from Twitter alone, about 10 million out of 500 million daily-published tweets are geo-tagged[1]. This has encouraged researchers to incorporate the spatial dimension of microblogs towards identifying the location of detected events.

The approach by Mathioudakis et al. [113] aims at finding the geographic locations exhibiting a burst in the frequency of extracted keywords. The area of interest is split based on a regular grid, and then, by minimizing a cost function, a subset of the cells are labeled as bursty and others as non-bursty. The technique proposed by Rattenbury et al. [127] aims at extracting place and event semantics from Flickr tags. The rationale behind their approach is that significant usage patterns for event and place tags should appear as a burst over a period of time or over parts of space. However, these approaches are based on batch processing and are not scalable for a large number of incoming microblogs.

Lee and Sumiya [100] present a system to detect unusual geo-social events by leveraging the *geographical regularities* inferred from the normal behavior patterns of geo-tagged microblogs. For this, they first partition the geographic space into regions of interest (ROI) using the K-mean clustering method [74] applied to the geographical coordinates (long/lat) of the collected microblogs. This type of clustering-based partitioning is used to better deal with heterogeneous regions differently. Then, for each region, a regular pattern is summarized from the following statics: (1) the number of microblogs, (2) the number of users inside the region, and (3) the number of users entering the region. The statistics of these patterns are accumulated over historical microblogs using a 6-hour time interval and called geographical regularity. The geographical regularity of each region is regarded as a baseline to statistically determine whether the current geographical regularity indicates the occurrence of a geo-social event at its respective region.

The approach by Walther and Kaisser [152] detect geo-spatial (localized) events by generating candidate event clusters based on the spatio-temporal proximity of tweets. This clustering mechanism is similar to DBSCAN [73], where the clustering module checks, for each new incoming microblog, whether there are more than x other

[1]http://www.idigitaltimes.com/twitter-statistics-2014-new-interactive-map-shows-where-people-tweet-and-how-muc h -398862, accessed Apr. 2015

microblogs published in the last y minutes in a radius of z meters. These parameters are set to 3, 30, and 200, respectively. The generated clusters are then examined to filter out non-event clusters using a supervised machine learning technique (see Section 2.6.1). Instead of conducting a spatial clustering at the microblog level, Boettcher and Lee [35] follow the feature-pivot detection and examine the locality of events at the word level. They built a framework that continuously extracts words W published during a time window from Twitter streams and generates subsets of length 1,2, and 3 words out of the powerset of W. Then, the DBSCAN clustering algorithm is employed to identify the subsets having limited spatial extent based on the geo-coordinates of the tweets in each potential event cluster. Finally, a classifier is trained on a set of labeled events to filter out non-event clusters.

For the purpose of detecting small-scaled localized events, the main challenge is the lack of microblogs that are geo-tagged. Hence, some efforts focus first on estimating the location of non-geo-tagged microblogs. In [151], the concept of virtual sensors are introduced, where each sensor is in charge of monitoring a specific geographical location. If an incoming tweet is not geo-tagged, it is passed on to a location extractor module that uses a local geo-coding service described in [150]. This geo-coder is basically built on top of both the GeoNames gazetteer[1] and Flickr data. Using this geo-coding service, an administrative hierarchy of geographic concepts is built to resolve ambiguity resulting from the existence of different places having the same name. After geo-coding, only geo-tagged tweets are mapped to their respective sensors. A sensor triggers the occurrence of an event at its respective location when the mapped tweets lead to changing the emotional state during a time window.

Watanabe et al. built a local event detection framework, called Jasmine [154]. It estimates the location of non-geotagged tweets to increase the chance of finding small-scaled localized events. For this, they built a database of <place name, geo-coordinate> pairs by making use of geo-tagged tweets. The non-geo-tagged tweets are examined, looking for phrases that match the place names in the database. When found, the corresponding geo-coordinates are assigned to that non-geo-tagged tweet. To ensure a reliable geo-tagging procedure, ambiguous place names (e.g., "McDonald's") that have a high-variance distribution of geo-coordinates are eliminated from the database. After geo-tagging, they search for place names and count the number of key terms co-occurring with each place name. However, their method fails to find the localized events when no place names pointing to the locations of these events are mentioned. On the other hand, Sankaranarayanan et al. [135] employ different sources

[1] `http://geonames.org/`

of spatial information to mitigate the sparsity of geo-tagged tweets. Their approach extracts news as clusters of news-related tweets. To enable spatial exploration of these news clusters, the geographic focus of each is quantified using both toponyms (place names) that are extracted from the content of tweets and the location metadata from the user profile.

Lee [98] uses a two-phase approach to mine spatio-temporal information from Twitter Streams. In phase-1, the *BursT* [99] framework is applied to dynamically assign a weight to each word in the tweets published during a sliding window. Then, the IncrementalDBSCAN [59] clustering mechanism is employed to group emerging topics in real-time. In phase-2, the spatial distribution of the bursty words in each cluster is analyzed to determine whether the cluster has a local or global spatial spread. However, *BursT* relies on the frequency statistics collected during a sliding window to distinguish significant (bursty) words from others. The main shortcoming of this approach is that it is inable to filter out words that are published frequently and repeatedly even at non-event scenarios, e.g., "lol", "job", and "morning". A solution to this problem is to use the discrepancy paradigm [96] that measures the deviation between the current usage frequency and its expected baseline. However, the advantage of this two-phase approach is that, at the beginning, the spatial dimension is ignored and non-bursty terms are excluded by utilizing the timestamps attached to the arriving messages. Then, the spatial dimension is considered only for bursty candidates. By this, the computational cost of analyzing the spatial information of non-bursty terms is avoided, which is crucial in the context of social media where huge and noisy content is arriving constantly. Likewise, Sakaki et al. [130] studied the temporal distribution of some predefined words. If such words show burstiness in usage, Particle and Kalman filters are applied to the geo-coordinates of event-related tweets in order to estimate the location of corresponding events.

Lappas et al. [96] presented a framework that aims at tracking the spatio-temporal burstiness of terms (words) using online settings. Two types of burstiness patterns are studied: 1) combinatorial patterns resulting from aggregating the bursty streams over an entire area of interest, and 2) regional patterns that refer to spatio-temporal patterns aggregated from spatially-close streams. It is hard to apply their approach to microblogging streams as microblogs are sparsely distributed over space, which might lead to generating many spurious bursts and making the maintenance of a baseline word usage for each stream intractable.

Budak et al. [39] introduced an algorithmic tool, called GeoScope, for detecting geo-trends from social networks by reporting trending location-topic (l, t) pairs. These

geo-trends are identified continuously using a sliding window approach. Geoscope offers theoretical guarantees for extracting all correlated pairs while requiring only sub-linear space and running time. However, in order to ensure scalability and to maintain a manageable list of locations at which trending topic-location pairs can be uncovered, only the locations that are at least θ-frequent in the current window are considered. That is, to track geo-trends at a certain location l_i (e.g., city), the number of pairs $|(l_i, .)|$ should occur at least $N\theta$ times where N is the number of elements in the time window and $\theta \in [0, 1]$. In this case, locations with frequencies below this lower bound are likely to be filtered out, which prevents uncovering trends with few occurrences.

Detecting Planned Events

A planned event is a scheduled event that people are aware of its topic, location, and time in advance, while unplanned events are those that occur suddenly, e.g., car accidents. Although people might know the nature and the consequences of unplanned events beforehand, they usually cannot predict when and where they will take place. To date, the majority of event detection approaches extract general event entities, i.e., entities for both planned or unplanned events. However, the detection of unplanned events is more challenging, in particular, in the context of NED detection tasks, where the timely detection of these events is important. This difficulty comes from the fact that only a few number of event-related microblogs are published close to the event's start time.

On the other hand, some approaches are tailored to detect planned events using the content of social media, which has proven to be useful in different application scenarios, e.g., harvesting the social opinions about events [159] and constructing event calenders [128].

For the sake of extracting planned events based on query-free detection, Ritter et al. [128] introduced TwiCal, a system for generating an open-domain calendar encompassing events extracted from a Twitter dataset. An NLP-based approach for finding named entities and event phrases from tweets is used. In addition, they exploit a probabilistic latent-variable model to divide the extracted events into categories.

The approach by Benson et al. [30] identifies tweets pertaining to concert events using a statistical model that annotates artist and venue words in Twitter messages. However, the detected events are limited to those with explicit artist and venue mentions. To overcome this limitation, Becker et al. [26] proposed a generic methodology for extracting and aggregating information about planned events. Event features

such as, title, description, time, and venue are retrieved from some event aggregation sites (e.g., Last.fm[1] and EventBrite[2]), and then these features are used as queries to extract event-related content from a number of social media sites, namely, Twitter, YouTube and Flickr.

2.7 Summary and Discussion

In this chapter, we introduced necessary concepts and techniques to extract event information from social media content. For this, we first presented the Topic Detection and Tracking program (TDT), as a first systematic work on the task of event detection and tracking. Then, we discussed basic concepts and models to process data streams, forming the basis of designing systems in support of the online mining of event information from a stream of documents. The main approaches on event detection and tracking from traditional media were introduced, and then we showed how these approaches have evolved and been adapted to suit the unique nature of social media content.

A surprisingly large number of works have been conducted on detecting events from social media. In this chapter, we have presented the most related efforts, classified based on the methodology adopted and the objective of the detection process. Recall that we consider, in this thesis, the near real-time detection and tracking of localized events. Therefore, while discussing existing approaches, we gave particular attention to those that are adapted to social media, process messages in an online fashion, and/or location-aware.

Existing approaches on event detection do not account for the spatial dimension of published microblogs while tracking already detected events. However, we argue that the role of this dimension in the tracking task is inevitable. Suppose, for example, two musical concerts are taking place at the same time. The microblogs published about both events hold the same general topic. As a result, this might lead to mapping these messages to one event or might mistakenly assign the microblogs of a certain event to the cluster of the other event, which is an incorrect event tracking. Of course, incorporating the spatial dimension will be a typical solution that tackle such a problem by providing a better spatial-centric clustering.

Recall that one of our main objectives in this work is to estimate the location of detected small-scale events using a fine-grained spatial resolution. In fact, it is

[1] `http://www.last.fm/`
[2] `https://www.eventbrite.de/`

a computationally infeasible process, especially when dealing with a huge stream of incoming microblogs. Knowing that the majority of microblogs are not related to real-world events, one solution to mitigate this is to eliminate non-event microblogs at an early stage before considering their spatial dimension.

For a reliable event extraction, some existing approaches employ probabilistic models to uncover latent event patterns from a corpus of documents. However, this line of research suffers from the following limitations when applied to social media content: (1) A fixed vocabulary set is assumed a priori. However, the content of social media is very dynamic and, at each single moment, a large number of new tokens (words) appear. (2) Fitting model parameters is computationally expensive, in particular when continuously including new arriving microblogs in the learning process. (3) Microblogs published on a small-scale event are usually very few, which can be easily underestimated by current probabilistic models.

By revealing the limitations of existing techniques, we aim in this thesis at building an event detection framework that combines the following important features. (1) Considering all dimensions of events, i.e., semantic, time, and location. (2) Tracking the evolution of localized events until they diminish by estimating a dynamic score that changes over time for each detected event. (3) Robust against noise and spatio-temporal problems arising when studying the spatio-temporal distribution of keywords. (4) Providing a good trade-off between accuracy and scalability.

Chapter 3

Keyword Extraction from Social Media

The online extraction of *keywords* is the first step towards the detection of ongoing real-world events and trends. Once keywords are extracted, events can then be formed from grouping semantically-related keywords based on some common features as discussed in Chapter 2. Keyword extraction is a well-established research area, where a huge body of research has been conducted to extract keywords from traditional text. However, for the streaming and noisy content of social media, the process of keyword extraction is still in its infancy and needs to confront a number of challenges.

In this chapter, we introduce KeyPicker, a framework to perform a near real-time extraction of keywords from a stream of microblogs. Different from existing keyword extraction techniques, KeyPicker has a number of features making it both reliable and practical for the noisy, heterogeneous, and dynamic nature of text in social media. These features are summarized as follows:

(1) KeyPicker is inherently incremental, that is, when the content of a microblog has been processed, it will not be considered again;

(2) It quantifies the significance of a word based on two characteristics: *burstiness* and *recurrence*, both of which are helpful in handling the noisy nature of social media content;

(3) To determine whether a word holds both characteristics, a variation of the discrepancy paradigm is developed to measure the deviation between the current usage pattern of a word and an expected baseline of it;

(4) The historical data, from which the baseline parameters are estimated, is updated and materialized to efficiently capture the dynamics of social media.

The experimental results show the effectiveness and efficiency of KeyPicker compared to baseline and state-of-the-art approaches in extracting keywords and tracking their evolution.

3.1 Introduction

Real-world events are the most influential stimuli that encourage users to publish via different microbloging services. One or more trending topics come into existence as a result of event occurrences. The words composing each trend are used more frequently during an event than in normal situations. For example, directly after an earthquake occurs, there is a remarkable surge in the number of related microblogs published by users [130]. As a consequence, event-specific words will exhibit some burstiness such as "OMG", "earthquake", and "shaking". We refer to these words as *keywords* for simplicity. The extraction of keywords from text streams is a broad research area that has been studied extensively, see, e.g., [62, 90, 113, 172]. Keyword extraction can be used, e.g., in building frameworks in support of real-world event detection [127, 156], emerging topic extraction [114], and context-aware search [96].

In the context of social media, however, the task of keyword extraction is non-trivial and faces many challenges. In this chapter, we tackle the following challenges:

1. *High arrival rate of microblogs.* The huge amount of incoming microblogs published through social media sites requires a scalable and single-pass procedure so that the large number of incoming microblogs can be processed efficiently.

2. *Noisy content.* As discussed in Chapter 2, the textual content of social media has a lot of typos, shortcuts, abbreviations etc. Moreover, a large percentage of published microblogs are pointless babble without event-related information[1]. Therefore, traditional content preprocessing methods, e.g., word stemming and in-dictionary word check, are not effective and will lead either to the loss of informative words or the incorporation of useless ones.

3. *Temporal outliers.* These outliers refer to event-related keywords mentioned before/after their respective events. Although such keywords describe the events

[1]http://www.pearanalytics.com/blog/2009/twitter-study-reveals-interesting-results-40-percent-pointles s-babble/, accessed Jan. 2015

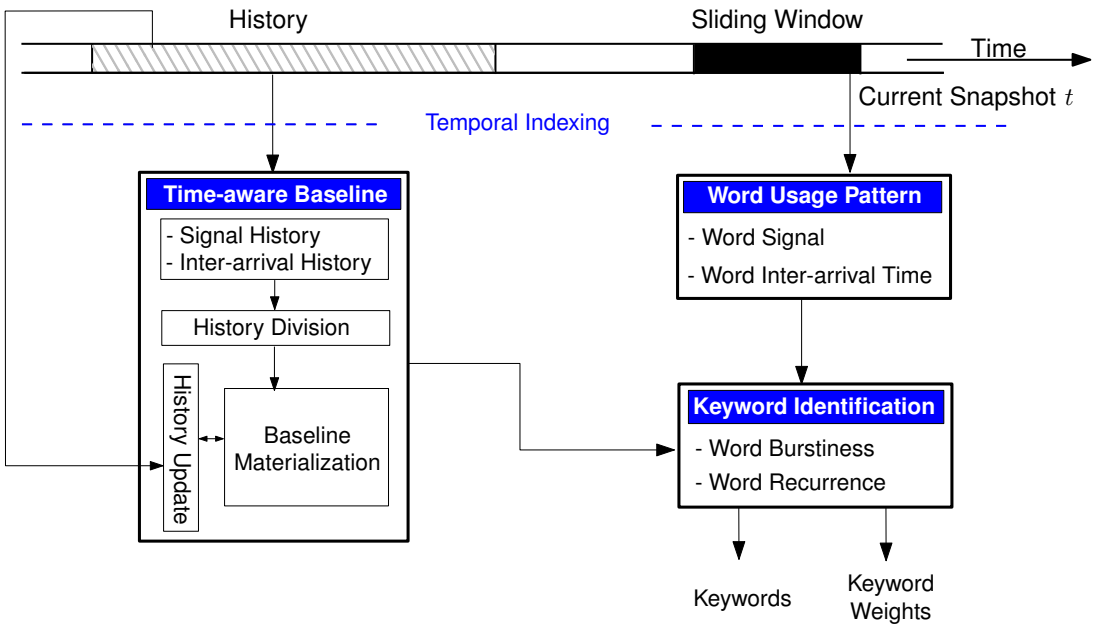


Figure 3.1: KeyPicker system overview.

thematically, they might prevent an accurate estimation of time-specific event information, e.g., the event's start and end time.

4. *Temporal heterogeneity.* The number of contributions of a certain word may substantially change based on the posting time. For instance, the word "morning" is used more frequently in the early morning than at noon.

Such challenges stands against a direct adoption of traditional data mining techniques that mainly rely on batch processing and perform well only on "formally-written" text.

Motivated by the aforementioned objectives and challenges, we introduce KeyPicker, an incremental approach to extract keywords from a stream of microblogs based on the concept of a sliding window. Figure 3.1 illustrates the main components of KeyPicker and the interaction between them. As time progresses and the window slides, KeyPicker digests incoming microblogs to extract new keywords or update the state of already extracted ones. For this, the usage pattern at the current snapshot t is quantified for each word. The words having abnormal usage patterns are then identified as keywords. To uncover the abnormality of the usage pattern of a word, a variation of the discrepancy paradigm that was proven to perform well in streaming scenarios is adopted [57]. It is based on measuring the deviation between the current usage pattern of a word and its expected usage baseline. The higher the deviation value, the higher the likelihood that the word is a keyword. The expected usage baseline of a certain word is estimated from its respective historical

observations. Moreover, a weight is estimated for each extracted keyword, reflecting the evolution of its significance over time. These weights will mainly be exploited to study the evolution of localized events, an aspect that will be discussed in more detail in Chapter 4.

The contributions of this chapter are summarized as follows:

(1) We organize the microblogs published during the sliding window based on a two-level index structure. This indexing scheme allows for a constant-time retrieval of count statistics pertaining to words contained in these microblogs.

(2) We utilize the change in word inter-arrivals observed in the most recent snapshots (within the sliding window) in order to mitigate the impact of temporal outliers and to remove noisy and rarely-mentioned words. That is, in addition to quantifying how bursty a word at a particular snapshot is, we also measure how recurrent it is in contiguous snapshots.

(3) We propose to estimate a time-aware usage baseline for each word observed in history in order to cope with the temporal heterogeneity of the usage patterns of words.

(4) To prevent baseline parameters from being outdated by time, we update the history by incrementally incorporating recent observations and eliminating expired ones. Moreover, we ensure both a sound incremental update and an efficient retrieval of usage baselines by aggregating and materializing the statistics that are sufficient to estimate such baselines.

This chapter is organized as follows. First, in Section 3.2, research efforts related to keyword extraction from social media are discussed. Then, we present the main notations, concepts, and problem statement in Section 3.3. In Section 3.4, we address the steps needed to preprocess and index microblogs before conducting the actual extraction of keywords. The concepts of word signal and baseline are introduced in Section 3.5, which will be used in Section 3.6 to quantify two keyword characteristics based on the discrepancy paradigm. The details of our datasets and experimental evaluation are presented in Section 3.7. Finally, we summarize this chapter and discuss some applications on keyword extraction from social media in Section 3.8.

3.2 Related Work

A large body of research efforts on extracting keywords from chronologically-ordered text documents has been and is undertaken. We can group these efforts into two categories:

Retrospective approaches. Here, the streaming nature of documents is neglected and they are processed in an offline fashion [62, 114, 138, 156, 172]. Unlike the traditional vector space model (VSM), a number of probabilistic models, e.g., LDA [34] and PLSA [78], are used to extract topics and rank their associated keywords. Dynamic topic models have been proposed as extensions of these probabilistic models by including the chronological sequence of documents in the learning process [33, 81]. This is helpful in capturing the temporal evolution of extracted topics. However, such models cannot process documents incrementally and extract keywords at an early stage of their corresponding trends. The burst model proposed by Kleinberg [90] has inspired a long sequence of related efforts. It models bursts using a finite state automaton, such that each state represents a certain arrival rate based on a Poisson arrival process. Then, finding the sequence of burst levels (states) is reduced to an optimization problem of minimizing a cost function on the given time series of word inter-arrivals. The complexity of this approach is high for large-scale analysis knowing that this optimization is performed for each encountered word individually.

Online approaches. In this category, the keyword extraction approaches account for the streaming nature of documents and try to extract keywords as soon as they appear. Guzman and Poblete [72] propose a scalable online approach based on windowing variation. Their approach consists of five modules that are divided into two main stages: 1) data preprocessing and 2) bursty keyword detection. In the first stage, the received tweets are filtered to choose only the ones written in some languages. Then, the chosen tweets are enqueued for posterior analysis, and finally, packed into disjoint time windows. In the second stage, the relevance variation rate is estimated for each keyword. The keywords having negative rates are discarded and the remaining ones are sorted based on their variation rates.

Other efforts adopt the discrepancy paradigm [43, 57] to compare the current behavior of a word against its historical usage pattern and identify those with abnormal rise in usage as keywords [3, 4, 95, 96]. Li et al. [103] built a framework to detect the existence of bursty segments in Twitter messages. A segment reflects a phrase that is a semantically-meaningful combination of words. First, segments are extracted from tweets using an optimization [104]. After that, bursty segments are identified within

a time window according to their frequencies. Sakaki et al. [130] have come up with a promising approach to detect the occurrence of certain events in real-time using some predefined keywords, which makes it inapplicable in detecting general bursty trends.

Lee et al. [99] attached a dynamic score combing the current usage frequency of a word with its arrival rate for each word observed in the stream. The higher the computed score, the more significant the word is. However, their approach, called *BursT*, assigns high score values to commonly used words, such as "morning" and "job". This behavior can be handled by making use of the discrepancy paradigm. On the other hand, the words of social media are temporally heterogeneous (see Section 3.5.3) in the sense that words have different usage patterns during a day.

He et al. [75] model bursts as dynamic phenomena borrowing concepts from physics, i.e., mass and velocity, and derive momentum, acceleration, and force from them. Although this methodology captures the dynamics of bursty topics well, it does not consider word usage heterogeneity and outliers that are inherent in social media content. Our framework KeyPicker accounts more for this noisy nature. To handle temporal outliers, when people mention keywords before/after their respective events, both keyword burstiness and inter-arrival are measured based on the discrepancy paradigm. For temporal heterogeneity, the baseline parameters are estimated from a history with periodic division. Moreover, the history of a word is updated and materialized periodically to ensure both reliability and efficiency in extracting keywords from social media.

3.3 Background and Problem Statement

In this section, first, we illustrate the temporal organization of microblogs along with the notations used to access these microblogs and the contained word statistics. A summary of these notations is described in Table 3.1. We conclude this section by presenting the problem statement.

3.3.1 Temporal Organization of Microblogs

The time dimension is inherent and very important when dealing with data streams. To cope with the huge amount of incoming microblogs, KeyPicker exploits a sliding window on the stream of microblogs and incrementally processes the most recently posted microblogs. The timeline is divided into *snapshots*.

Table 3.1: Main notations used for extracting keywords and estimating a dynamic weight for each.

Notation	Description
t	current snapshot
M^t	microblogs published at t, m is a microblog in M^t
$\mathcal{W}$	sliding window
$\mathcal{W}^t$	set of words mentioned at snapshot t
$\mathcal{W}^t_w$	microblogs published at t and contains word w
c	sliding window size (number of snapshots in $\mathcal{W}$)
s^t_w	signal of word w at snapshot t
a^t_w	inter-arrival time of word w at snapshot t
S_w	signal history of word w
I_w	inter-arrival time history of word w
v	history length
q	periodic interval length
$\mathcal{B}$	materialized usage baseline
K^t	keywords extracted at t

Definition 3.1 (Snapshot) *Given a sequence $(\cdots, t-2, t-1, t)$ of fixed-length, adjacent, and non-overlapping time intervals composing the timeline. Each such interval is called "snapshot" and corresponds to the time interval during which a number of microblogs is published. The snapshot t is assumed to be the current snapshot.*

These snapshots are the main building blocks over which a *sliding window* $\mathcal{W}$ moves. The granularity of these snapshots determines how fast the time window $\mathcal{W}$ slides. The snapshot length can be set to a fine resolution, e.g., one minute, to achieve online or near real-time processing of microblogs.

Definition 3.2 (Sliding window) *The last c snapshots form a sliding window $\mathcal{W}$ holding the most recent microblogs to be processed.*

Let us denote the set of microblogs published at t by M^t. Each microblog $m \in M^t$ consists of a number of fields.

Definition 3.3 (Microblog) *A microblog $m = (S, id, uid, loc, time)$ consists of a set of words (S), a microblog identifier (id), a user identifier (uid), a creation time ($time$), and geo-coordinates (loc).*

In this chapter, we make use of all these fields except *loc*, which will be considered in Chapter 4.

This sliding-window-based processing paradigm is crucial when handling large volumes of incoming microblogs. For this, processing the microblogs published during the most recent snapshot t starts directly when the end time point of snapshot t is reached. The next processing iteration will be triggered at $t+1$ to analyze microblogs within M^{t+1}, and so on. For efficiency reasons, we implement the sliding window $\mathcal{W}$

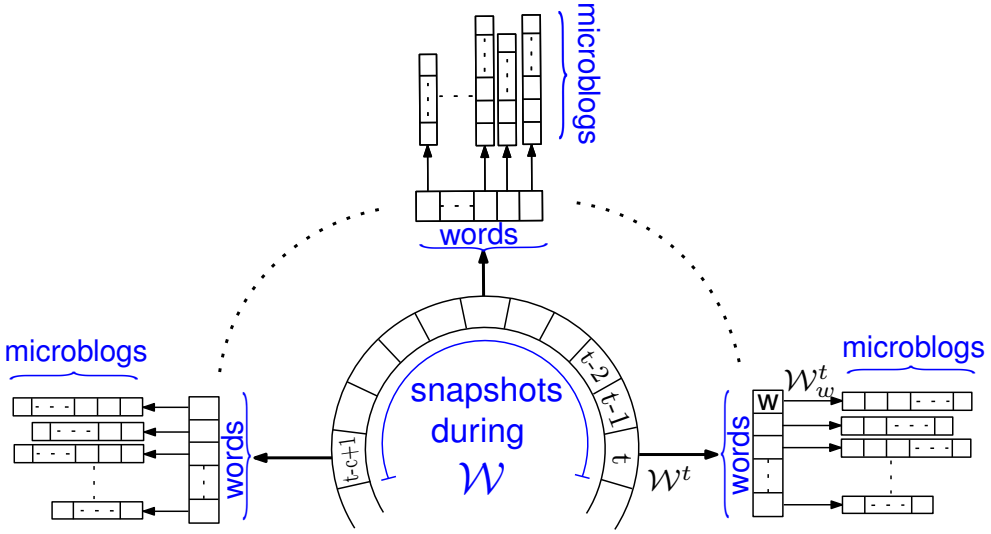


Figure 3.2: The temporal index of the microblogs published during $\mathcal{W}$. The key $\mathcal{W}^t$ of the inverted index $\mathcal{W}$ points to a set of words. These words are also keys for another inverted index $\mathcal{W}^t$. The postings of the key $\mathcal{W}^t_w$ are the ids of microblogs published at snapshots t and containing word w.

as an inverted index such that each microblog within $\mathcal{W}$ is indexed by the snapshot during which it is published. For example, to directly access the microblogs published during snapshot t, the time index $\mathcal{W}^t$ is used. Moreover, we add another word-level of indexing. In other words, the microblogs published during a certain snapshot t are maintained in an inverted index accessed by $\mathcal{W}^t$. As can be seen in Figure 3.2, the keys of this index are the words $\{w \in m.S | m.time \in t\}$ and the postings are the ids of microblogs containing these words. The microblogs published during snapshot t and containing word w are denoted by $\mathcal{W}^t_w$. Therefore, constant time is required to access the microblogs published during a certain snapshot and containing a certain word.

3.3.2 Problem Statement

After a new snapshot t elapses, *a set of keywords K^t is to be extracted* from the content of microblogs published during t, such that $K^t \subseteq \mathcal{W}^t$. For each $k \in K^t$, *a*

dynamic weight, denoted $weight(k, t)$, is computed to reflect how the significance of k evolves over time.

To judge whether a certain word is a keyword or not, its current usage behavior needs to be compared against an expected baseline. This baseline is represented as a set of parameters that are estimated from historical observations. Hence, the task is *to estimate, for each word w, at each snapshot t, a time-dependent baseline*. However, the content of social media has a dynamic nature such that, by time, new words appear and others diminish. Therefore, *the computed baseline should be maintained and updated over time*. Moreover, a baseline of a single word might be requested several times during a short period of time, and thus, appropriately *materializing corresponding baselines* should reduce the computation time substantially.

3.4 Temporal Index Update

In Section 3.3.1, we described the structure of the temporal index that holds and indexes the microblogs' content published during the sliding window $\mathcal{W}$. As time progresses and the new snapshot t elapses, there is a need to include the recently arrived content and to remove the content of the expired snapshot. Hence, in this section, we describe the steps needed to update the temporal index $\mathcal{W}$, which includes two main operations:

1. Inserting the content of microblogs published during snapshot t. First, text preprocessing is conducted for each microblog to retrieve words representing the postings of index $\mathcal{W}^t$. Then, the microblogs are stored according to the words they contain as illustrated in Section 3.3.1.

2. Deleting the content of the microblogs published during the expired snapshot $t - c$.

Algorithm 3.1 shows the steps needed to preprocess the content of microblogs M^t published during the current snapshot t and to update the index $\mathcal{W}$ accordingly. First, in Line 1, a new inverted index $\mathcal{W}^t$ is generated, maintaining the microblogs published during the current snapshot t, and then, $\mathcal{W}^t$ is appended to $\mathcal{W}$. Before inserting word entries in the index $\mathcal{W}^t$, some preprocessing steps are performed to cope with the noisy nature of the content of microblogs. In fact, microblogs contain a lot of spelling errors, colloquial expressions, shortcuts etc. Moreover, some social media sites, e.g., Twitter, impose a size limitation on the length of a microblog's

content, which forces users to use shortcuts to convey information. In addition, the vocabulary size of the textual content published through social media is virtually unlimited; and therefore, content preprocessing is a must. In our framework, we apply the following preprocessing operations for each microblog arriving during the recent snapshot t:

1. Per-user aggregation of microblogs (Line 2): In the context of social media, one should be careful when counting words, because some microblogs are republished several times for the sake of news distribution, advertisements etc. Multiple occurrences of such microblogs do not necessarily reflect a certain event. To deal with such cases, the microblogs posted by a single user at snapshot t are aggregated to form just one microblog.

2. discard special characters (e.g., #, ?, !, ”). (Line 6)

3. remove stop words, words having a length less than three, and URLs. (Line 7)

Words whose lengths are less than 3 are disregarded because they are usually uninformative, e.g., smileys [94]. Furthermore, unlike traditional preprocessing techniques, we do not use stemming as stemmers are not effective when the text stream is flooded with a lot of typos and colloquial words. On the other hand, stemmers are language-specific, which implies the need for a separate stemmer for each language.

The preprocessed version *tmp* of word w is then used to update $\mathcal{W}^t$ as can be seen in Lines 8-10. First, we check whether $\mathcal{W}^t$ contains the posting *tmp*, and if not, *tmp* is inserted and a reference to this posting is returned as $\mathcal{W}^t_{tmp}$ (Line 9). Then, microblog m is added to the list of microblogs published during t and containing word *tmp*, namely the list $\mathcal{W}^t_{tmp}$. Finally, in Line 11 the memory space reserved to maintain the content of microblogs posted during the expired snapshot $t - c$ is de-allocated to free space for new microblogs arriving at subsequent snapshots.

3.5 Word Usage Pattern and Baseline

In this section, we study the usage pattern and baseline of each word $w \in \mathcal{W}^t$. The usage pattern describes the behavior of the word at the current snapshot t (Section 3.5.1). This behavior is represented through two estimates: *word signal* and *inter-arrival time*. In Section 3.5.2, we study the expected behavior of a word, called *word baseline*, which will be estimated from historical observations. Both the usage

Algorithm 3.1: Temporal index generation and content preprocessing.

Input: Index $\mathcal{W}$, Microblogs M published during t, sliding window size c
Output: $\mathcal{W}^t$ inverted index appended to $\mathcal{W}$
// Index $\mathcal{W}^t$ instantiation
1 $\mathcal{W}^t \leftarrow \mathcal{W}.appendIndex(t)$
2 $M \leftarrow aggregateMicroblogsPerUser(M)$
3 **foreach** $m \in M$ **do**
4 **foreach** $w \in m.S$ **do**
5 $tmp \leftarrow w$ // tmp holds a preprocessed version of w
6 $tmp \leftarrow tmp.omitSpecialCharacters()$ // $\#, !, ?, “, \ldots$
7 **if** $!tmp.isStopWord()$ & $len(tmp) > 2$ & $!tmp.isURL()$ **then**
8 **if** $tmp \notin \mathcal{W}^t$ **then**
9 $\mathcal{W}^t_{tmp} \leftarrow \mathcal{W}^t.appendIndex(tmp)$
10 $\mathcal{W}^t_{tmp}.insert(m)$
// remove content of expired snapshot
11 $\mathcal{W}.deleteIndex(t - c)$
12 **return** $\mathcal{W}$

pattern and baseline are then exploited in Section 3.6 to determine whether w is a keyword and to compute its dynamic weight as well.

3.5.1 Word Usage Pattern

Assume that a certain event is taking place at snapshot t. Therefore, some related words will exhibit a surge in usage at t and this bursty nature might last for some time, i.e., for a number of snapshots. To better describe the current usage behavior of a word, we account for not only how frequent this word is used at snapshot t but also how often it appeared at previous snapshots within $\mathcal{W}$. For this, the *usage pattern* of a word is quantified using two estimates:

(1) *Word Signal*: It represents a quantitative measure reflecting the relative frequency of word w, which is reported at the end of snapshot t. The higher the word signal, denoted s^t_w, the more users mention w at t.

Definition 3.4 (Word Signal) *The signal of word w at snapshot t is the ratio between the number of microblogs containing w and the total number of microblogs published during the sliding window $\mathcal{W}$, defined as*

$$s^t_w := \frac{\sum_{i=t}^{t-c+1} |\mathcal{W}^i_w|}{\sum_{i=t}^{t-c+1} |M^i|}. \tag{3.1}$$

If some word w pertains to a certain event taking place at snapshot t, the signal of w is expected to have a positive value larger than in normal situations.

(2) *Word Inter-arrival*: The inter-arrival time[1] of word w at snapshot t is a good indicator of how temporally-close the occurrence of word w at time t is to its previous occurrence during $\mathcal{W}$. The closer the recurrences of word w at contiguous snapshots, the higher the evidence that the word relates to a certain event.

Definition 3.5 (Word Inter-arrival) *The inter-arrival time a_w^t of word w at snapshot t is the number of snapshots the current occurrence of that word is temporally apart from the preceding occurrence during $\mathcal{W}$.*

Formally, the inter-arrival time a_w^t of word w at snapshot t is defined as

$$a_w^t := \begin{cases} 0 & \text{if } s_w^i = 0 \ \ \forall i \in Q \\ t - \underset{i \in Q \wedge s_w^i > 0}{\arg\max}\, i & otherwise \end{cases} \tag{3.2}$$

where $Q = \{t-c+1, \ldots, t-1\}$ is the set of snapshots in window $\mathcal{W}$ without snapshot t. When word w appears only during the current snapshot t (case 1 in Eq. 3.2), the inter-arrival time a_w^t is set to zero. Otherwise, it is set to the difference between the current snapshot t and the closest one during which w appeared (case 2 in Eq. 3.2).

Example 3.1 *Assume that the word signals of word w during a 6-hour window are: $s_w^{t-5} = 0.2$, $s_w^{t-4} = 0.3$, $s_w^{t-3} = 0$, $s_w^{t-2} = 0.1$, $s_w^{t-1} = 0$, and $s_w^t = 0.2$. Then, the current word inter-arrival a_w^t is 2. This is because the first snapshot that has a positive signal before the current snapshot t is $t-2$, and thus, $a_w^t = t - (t-2) = 2$.*

3.5.2 Time-aware Usage Baseline

In order to identify a word w as a keyword at snapshot t, its current usage pattern discussed in the previous section should show an abnormal behavior. More precisely, s_w^t and a_w^t have to hold unusually high and small values, respectively. This abnormal usage pattern can be uncovered when compared to a *usage baseline* describing the word's non-bursty behavior. The usage baseline of word w is therefore a property essential to determine whether w is considered a keyword. A reliable estimation of such a baseline is important to obtain a high-quality keyword extraction procedure.

[1] inter-arrival for short

In the following, we first show how to quantify word baselines, address the problem of temporal heterogeneity, and finally discuss how to estimate time-aware baselines to cope with this problem.

Baseline Parameters

The baseline of a certain word is basically estimated using observations collected from a historical window (*history* for short) preceding the sliding window $\mathcal{W}$ (see Figure 3.1). This baseline is quantified as *baseline parameters* describing the probability distributions underlying these historical observations. In the following, we present what type of baseline parameters are required and how to estimate them.

Suppose that the microblogs collected during history are divided into a chronologically-ordered sequence of sets of microblogs $(M^0, M^2, M^3, \cdots, M^{v-1})$, where v is the history length (number of snapshots) and is set according to the inequality $v \leq (t - c + 1)$. Using this inequality, history does not overlap with the sliding window, yet a *gap* is allowed between history and the sliding window $\mathcal{W}$. This *gap* imposes a certain level of independence, and thus, lowers the correlation between historical observations and those corresponding to the sliding window.

To efficiently obtain a baseline for each word from history, the microblogs are indexed based on a temporal index structure similar to the one described in Section 3.3.1. We apply Algorithm 3.1 to create inverted indexes $\mathcal{W}^i$ for each historical snapshot $i \in \{0, 1, 2, \cdots, v-1\}$[1]. By this, we can retrieve count statistics for each word w in history in constant time. Using the method discussed in Section 3.5.1, two types of observations are extracted from these historical snapshots:

(1) Historical signals S_w: a sequence of chronologically-ordered signals of word w, i.e., $S_w := (s_w^0, s_w^1, \cdots, s_w^{v-1})$. We assume that these signals are generated from a normal distribution,

$$S_w \sim \mathcal{N}(\mu_w, \sigma_w),$$

which is determined by two parameters: the mean (μ_w), and the standard deviation (σ_w).

(2) Historical inter-arrivals I_w: a sequence of chronologically-ordered inter-arrivals of word w, which indicate the temporal gaps between consecutive positive signals

[1]The colon in the superscript $(0 : v-1)$ represents a sequence of snapshots from 0 to $v-1$, i.e., $\mathcal{W}^{0:v-1} = (\mathcal{W}^0, \mathcal{W}^1, \cdots, \mathcal{W}^{v-1})$.

in S_w. These inter-arrivals are assumed to be exponentially distributed,

$$I_w \sim Exp(\lambda_w),$$

where the inter-arrival rate λ is the only parameter to be estimated. The exponential distribution is usually used to describe the lengths of the inter-arrival times in a homogeneous Poisson process [130].

The aforementioned parameters (μ_w,σ_w,λ_w) are the baseline parameters of word w and estimated using maximum likelihood [31].

Temporal Heterogeneity and History Division

The usage patterns of the majority of words in social media suffer from *temporal heterogeneity* in the sense that the observations in S_w and I_w vary over time exhibiting some periodic usage patterns. This is mainly due to two main reasons:

(1) A large number of words are inherently periodic. For example, the word "morning" is frequently mentioned in the early morning, and then its usage frequency keeps decreasing until it approaches zero at night. This heterogeneous usage pattern recurs the following day, and so forth.

(2) The number of microblogs depends highly on the snapshot during which they are published. Usually, there are many more messages published during day time than during night.

Thus, the computed baseline parameters of word w, considering history in its entirety, can over- or under-estimate the real value depending on the snapshot used to determine a baseline. For instance, a small increase in the frequency of the word "car" published as a response to a certain event taking place during night might be erroneously ignored. This is because the word "car" is a common word, mentioned in different contexts, and thus, has a relatively high mean μ_w, in particular, at day time. Therefore, the estimated baseline for word w at the current snapshot t should be *time-aware*. That is, if t refers to a time point in the morning, then the baseline parameters have to be estimated from historical snapshots in the mornings of each day in history.

To handle temporal heterogeneity, to capture the periodicity of a signal, and hence to estimate a time-aware baseline, we divide the history of each word into a number

of non-overlapping, consecutive, and fixed-length *periodic intervals*, i.e.,

$$\mathcal{W}_w^{0:v-1} = (\mathcal{W}_w^{0:q-1}, \mathcal{W}_w^{q:2q-1}, \cdots, \mathcal{W}_w^{v-q:v-1})$$

where q is the interval length, i.e., the number of snapshots composing this interval. The rationale behind this division is the fact that snapshots corresponding to different

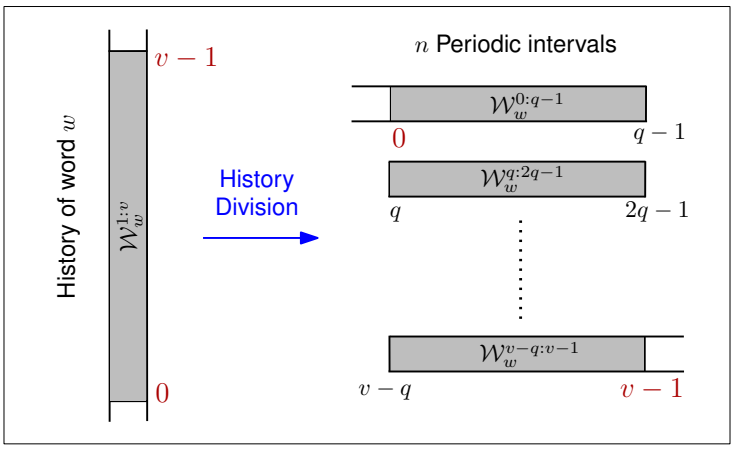


Figure 3.3: Dividing history into n fixed-length periodic intervals.

periodic intervals and having the same relative offsets refer to the same timing. For instance, if the ith snapshot in a certain periodic interval corresponds to 9am, then the ith snapshot of other intervals refer also to the same time, i.e., 9am. The history length (v) is chosen to be a multiple of q, i.e., $\frac{v}{q} = n \in \mathbb{N}^+$, where n is the number of periodic intervals (see Figure 3.3).

For simplicity, we assume that the majority of words have a sequence pattern that recurs daily. Thus, each periodic interval covers one day from history. In other words, if a snapshot of size 1 hour is used, then the length of the daily periodic interval q is 24 snapshots. Similarly, if one-minute snapshots are used, then the daily periodic interval is $24 \times 60 = 1440$ snapshots.

Time-aware Baseline

After conducting history division, the next step is to select a subset of snapshots from each periodic interval and use them to reach our target of estimating the time-aware baseline parameters denoted μ_w^t, σ_w^t, and λ_w^t for each word w at the current snapshot t. For this purpose, both sequences S_w and I_w undergo the history division described above. Then, a number of observations are chosen from each periodic inter-

val (see Figure 3.4). The snapshots of these chosen observations should be *temporally consistent* with the snapshots of the sliding window $\mathcal{W}$.

Definition 3.6 (Temporally-consistent snapshots.) *A snapshot m_1 from the sliding window is said to be temporally consistent with a snapshot m_2 from a periodic interval if and only if m_1 and m_2 are congruent modulo q, i.e., $m_1 \equiv m_2 \pmod{q}$.*

This congruence relation [129] guarantees that the chosen historical observations from each periodic interval and those observed in the sliding window have the same daily timing.

Example 3.2 *Assume that a 1-hour-snapshot sliding window of size 3 is used. If we want to estimate the time-aware baseline of the word "sun" at snapshot 13:00, the snapshots (11:00, 12:00, 13:00) are chosen from each periodic interval.*

Moreover, this congruence relation allows for an efficient access to historical observations in each periodic interval, which are temporally-consistent with the snapshots of the sliding window $\mathcal{W}$. To access such historical observations, their indexes are calculated using

$$indexes := \{(i-1)q + (j \bmod q)\},$$

for each periodic interval $i \in \{0, 1, \cdots, n-1\}$, and for each sliding window snapshot $j \in \{t-c+1, t-c+2, \cdots, t\}$. Therefore, the chosen historical signals are $S_w^t := S_w[indexes]$ that are used to estimate the time-aware parameters μ_w^t, σ_w^t. Similarly, the time-aware inter-arrival rate λ_w^t is estimated from the chosen historical inter-arrivals $I_w^t := I_w[indexes]$. Since both S_w^t and I_w^t are chosen from historical snapshots that are temporally-consistent with the sliding window, their corresponding distributions have less variability than the distribution of the entire set of observations, which helps mitigate the temporal heterogeneity problem.

3.5.3 History Update and Materialization

History Update

The vocabulary size of social media is virtually unlimited and the words mentioned in published microblogs have dynamic usage patterns. For example, in Twitter, a large number of hashtags appear everyday and others diminish after being used for a while. Furthermore, the usage pattern of some words changes over time. In other words, a word may exhibit a surge in usage as a result of a certain event. This increase in usage

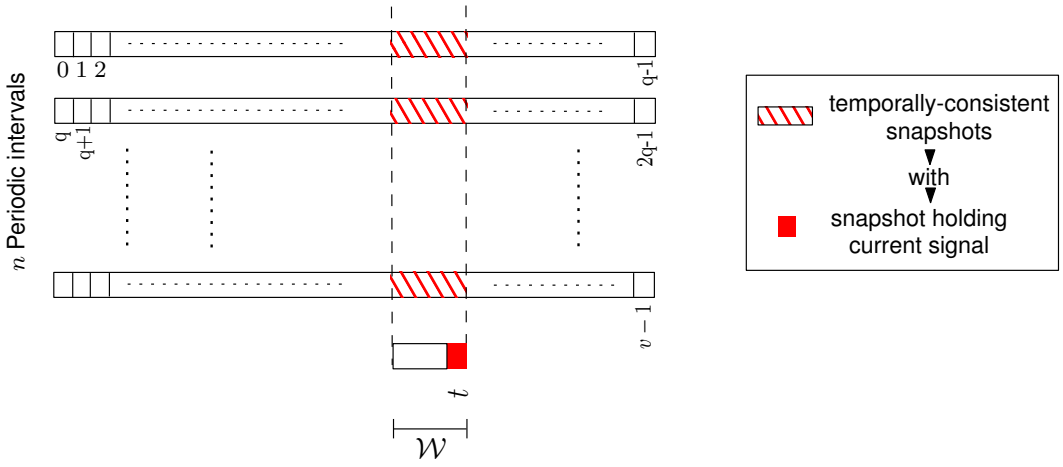


Figure 3.4: The time-aware baseline of word w at snapshot t is estimated from observations associated with temporally-consistent snapshots, indicated as diagonally-dashed snapshots.

might last for a long time and even after the event ends. As a consequence, the process of conducting history update and maintenance over time is essential, ensuring that the history includes the new words, getting rid of disappearing ones, and perfectly matching the current usage patterns of existing words.

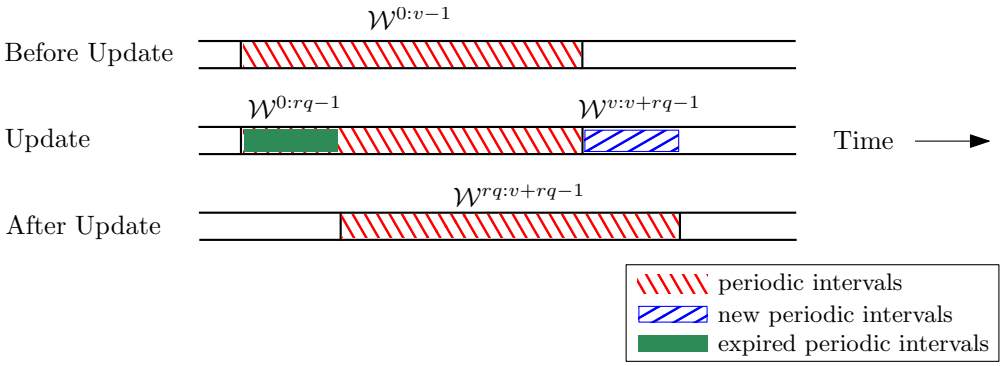


Figure 3.5: Updating history by appending r new periodic intervals, each of which is of size q, and by removing the same number of expired (oldest) ones. The resulting historical snapshots after update are $\mathcal{W}^{rq:v+rq-1}$

Updating the history requires the deletion of expired periodic intervals and the insertion of new and subsequent ones. As can be inferred from Figure 3.5, by updating the history of w, the system can now capture its usage dynamics by pushing its corresponding history forward covering more recent snapshots.

The history update procedure is triggered by observing the *gap* between history and $\mathcal{W}$ (see Figure 3.6). Initially, the *gap* is set to be a multiple of the length of a periodic interval q, meaning that the *gap* is the number of periodic intervals separating

the sliding window from history. It is not necessary to update history each time a new snapshot elapses. In fact, this update procedure depends on the dynamics of social media content and can be performed once every $r \times q$ snapshots, where $r \in \mathbb{N}^+$. In this case, the *gap* between the history and the sliding window remains the same over time and its original length is restored every r periodic intervals. If r is set to 1, then the history is updated each time a new periodic interval elapses, which is the fastest update rate that can be achieved. Expired periodic intervals, new ones, and the updated history are denoted $\mathcal{W}^{0:rq-1}$, $\mathcal{W}^{v:v+rq-1}$, and $\mathcal{W}^{rq:v+rq-1}$, respectively (see Figure 3.5).

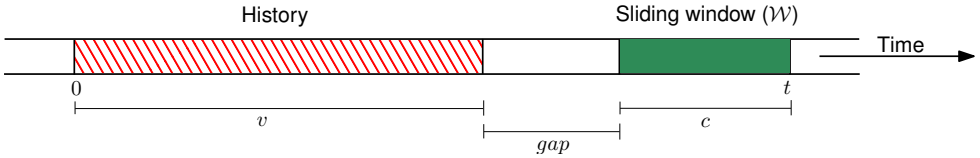


Figure 3.6: Historical window covering snapshots $0, 1, \cdots, v-1$ (diagonally-dashed filling), sliding window $\mathcal{W}$ (solid filling). History is separated from the sliding window by a *gap*.

Baseline Materialization

Recall that, for each word w mentioned at snapshot t, the baseline parameters $(\mu_w^t, \sigma_w^t, \lambda_w^t)$ need to be estimated on the fly using the historical observations S_w^t and I_w^t. This estimation process is repeated as long as w is being mentioned in subsequent snapshots, which requires a time complexity of $O(|\mathcal{W}^t| \times q)$, where q is the periodic interval length and $|\mathcal{W}^t|$ is the number of words at snapshot t.

To expedite this baseline estimation process, one solution is to materialize the parameters $(\mu_w^i, \sigma_w^i, \lambda_w^i)$ for each word in history at each *periodic snapshot* $i \in \{0, 1, \cdots, q-1\}$.

Definition 3.7 (Periodic snapshots) *These snapshots correspond to all possible time points at which the time-aware baseline parameters need to be estimated. Once a new snapshot t elapses, the corresponding periodic snapshot is calculated using the congruence relation described in Definition 3.6. Periodic snapshots are referenced using the indexes $\{0, 1, \cdots, q-1\}$.*

Example 3.3 *Assume that the periodic interval length is $q = 24$ and snapshot t is 244, then the periodic snapshot is 244 mod 24 = 4.*

Baseline parameters can be stored in a matrix-like data structure where the rows represent words and the columns are the periodic snapshots. Each entry of this matrix maintains the parameters (μ, σ, λ) for its corresponding word and snapshot. Although this type of materialization allows for a direct access to baseline parameters, it requires a relatively large maintenance time, in particular, when the history is to be updated. This is because the entire parameter estimation procedure has to be performed again, accounting for new periodic intervals and excluding expired ones. For this, we choose instead to materialize *synopses* (summaries) of the word signals, which are then used to estimate the baseline parameters in constant time. These synopses include count statistics and *sufficient statistics* required to compute the parameters (μ, σ, λ). The sufficiency is a well-known data reduction principle in statistics, stating that observations can be reduced with statistics whose use involves no loss of information, in the context of estimating unknown parameters for the probabilistic model underlying these observations [31]. In other words, if the value of a sufficient statistic is known, the observations do not contain any further information for estimating the model parameters.

The synopses maintained to estimate both μ_w^t and σ_w^t are:

- x_{sum} : the sum of historical signals in S_w^t.
- x_{sum}^2 : the sum of squared historical signals in S_w^t.
- x_{num}: the number of observations in S_w^t.

However, the synopses used to estimate the third baseline parameter λ_w^t are:

- i_{sum}: the sum of inter-arrivals in I_w^t.
- i_{num}: the length of I_w^t.

These five synopses for both S_w^t and I_w^t, which are sufficient to estimate the baseline parameters $(\mu_w^t, \sigma_w^t, \lambda_w^t)$, are referred to as *materialized usage baseline*.

Definition 3.8 (Materialized usage baseline) *The materialized usage baseline of word w mentioned during snapshot t, denoted $\mathcal{B}_w^t$, is defined as:*

$$\mathcal{B}_w^t := (x_{sum}, x_{sum}^2, x_{num}, i_{sum}, i_{num}) \tag{3.3}$$

The synopses of each word in history at each periodic snapshot $\{0, \cdots, q-1\}$ are maintained in a matrix referred to as $\mathcal{B}$ map as can be seen in Figure 3.7.

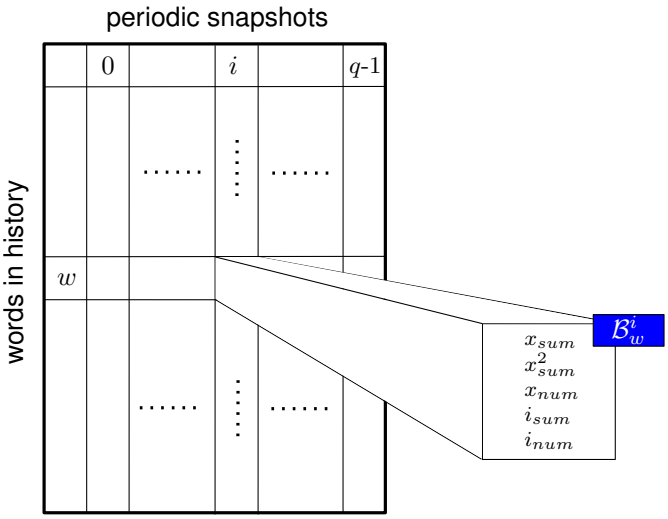


Figure 3.7: Materialized usage baseline map $\mathcal{B}$, maintaining synopses used to estimate the time-aware baseline parameters for each word in history.

Once history update is triggered, the synopses maintained in $\mathcal{B}$ are updated based on the following two operations:

(1) **Insertion**: accumulating the count statistics of words, observed in the new periodic intervals, in their corresponding entries in the $\mathcal{B}$ map (see Algorithm 3.2);

(2) **Deletion**: subtracting the count statistics of words mentioned in the expired periodic intervals from their respective entries in $\mathcal{B}$ (see Algorithm 3.3).

Algorithm 3.2 details the steps required for the insertion operation. In Line 3, if a previously unseen word w is encountered, a new row is created for w and appended to the end of the $\mathcal{B}$ map. Then, we iterate over each snapshot j within the new r periodic intervals (Lines 4-9). In Line 5, and before collecting statistics from snapshot j, its respective periodic snapshot z is identified using the congruence relation described in Definition 3.6. Then, the synopses corresponding to the new snapshot j are computed and accumulated in existing statistics pertaining to word w at the periodic snapshot z (Lines 9-9). Note that the inner loop is used to iterate over the snapshots that are temporally-consistent with those of window $\mathcal{W}^z$. The computed synopses from observations corresponding to these snapshots are added to the map entry $\mathcal{B}_w^z$.

Algorithm 3.3 is invoked directly after Algorithm 3.2 to eliminate the statistics of the expired periodic intervals $\mathcal{W}^{0:rq-1}$. Notice that these statistics are subtracted from their respective materialized baselines, ensuring an incremental update for these baselines. When no signal for word w is observed at any snapshot in history, i.e., $(\mathcal{B}_w^z.x_{num} = 0) \quad \forall z \in \{0, 1, \cdots, q-1\}$, the entries of this word are removed from

Algorithm 3.2: $\mathcal{B}$ Map Update (Insertion).

Input: new periodic intervals $\mathcal{W}^{v:v+rq-1}$, window size c, map $\mathcal{B}$, history size v, number of inserted periodic intervals r.
Output: Updated $\mathcal{B}$.

```
1  foreach w ∈ W^{v:v+rq−1} do
2      if w ∉ B then
3          B.insert(w)
4      for j ← v to v + rq − 1 step 1 do
5          selected z ← j mod q
6          if z = 0 then
7              // To avoid inter-arrivals from different intervals
8              lastIndex ← −1
9          for i ← (c − 1) to 0 step (−1) do
10             B_w^z.x_sum += S_w[j − i]
11             B_w^z.x_sum^2 += (S_w[j − i])^2
12             B_w^z.x_num += 1
13             if S_w[j − i] > 0 then
14                 if lastIndex ≠ −1 then
15                     B_w^z.i_sum += i − lastIndex
16                     B_w^z.i_num += 1
17                 lastIndex = i
18
19 return B
```

memory (Line 18). The process of accumulating statistics from the new periodic intervals (Algorithm 3.2) is deliberately performed before removing those from the expired intervals (Algorithm 3.3) in order to decrease the likelihood of removing a materialized baseline from memory and then re-instantiating it again when new signals appear directly in subsequent periodic intervals.

Now, when the time-aware baseline parameters μ_w^t, σ_w^t, and λ_w^t of word w at snapshot t are needed, it can be estimated in constant time, making use of its corresponding statistics maintained in the materialized baseline $\mathcal{B}_w^t$. Formally,

$$\mu_w^t = \frac{\mathcal{B}_w^t.x_{sum}}{\mathcal{B}_w^t.x_{num}}. \tag{3.4}$$

$$\sigma_w^t = \sqrt{\frac{\mathcal{B}_w^t.x_{sum}^2}{\mathcal{B}_w^t.x_{num}} - (\mu_w^t)^2}. \tag{3.5}$$

Algorithm 3.3: $\mathcal{B}$ Map Update (Deletion).

Input: expired periodic intervals $\mathcal{W}^{0:rq-1}$, window size c, map $\mathcal{B}$, history size v, number of removed periodic intervals r
Output: Updated $\mathcal{B}$.

```
1  foreach w ∈ W^{0:rq−1} do
2      for j ← 0 to rq − 1 step 1 do
3          z ← j mod q
4          if z = 0 then
5              // To avoid inter-arrivals from different intervals
6              lastIndex ← −1
7          for i ← (c − 1) to 0 step (−1) do
8              B^z_in.x_sum −= S_w[j − i]
9              B^z_w.x^2_sum −= (S_w[j − i])^2
10             B^z_w.x_num −= 1
11             if S_w[j − i] > 0 then
12                 if lastIndex ≠ −1 then
13                     B^z_w.i_sum −= i − lastIndex
14                     B^z_w.i_num −= 1
15                 lastIndex = i
16     // if no signal is observed for w in history
17     if sum([B^{0:q−1}_w].x_num) = 0 then
18         B.delete(w)
19 return B
```

$$\lambda_w^t = \frac{\mathcal{B}_w^t.i_{num}}{\mathcal{B}_w^t.i_{sum}}. \tag{3.6}$$

In the following section, we show how KeyPicker exploits these baseline parameters to identify event-related words (keywords) at the end of an elapsed snapshot.

3.6 Keyword Identification

To identify keywords among the huge stream of incoming words, we first describe two characteristics important to uncover the (current) significance of a word: (1) word *burstiness*, and (2) word *recurrence*. Then, we will show how we utilizes these characteristics along with the time-aware baseline discussed in the previous section to extract keywords and to estimate a dynamic weight for each.

3.6.1 Word Burstiness

A word used by a larger number of users than in normal situations during a certain period of time is referred to as *bursty word*. In the context of social media, the discrepancy paradigm is used [4, 96] in extracting bursty words due to its efficiency in incrementally processing words contained in the incoming stream of microblogs. It measures the deviation between a current usage frequency of a word w and its expected baseline. If this deviation exceeds a certain threshold, the word is considered bursty. We apply a variation of the discrepancy paradigm, which uses the z-score measure to identify bursty words. Based on the assumption that word signals are normally distributed (see Section 3.5.2), a burstiness degree is computed for each word w at snapshot t as

$$burst(w,t) := \frac{s_w^t - \mu_w^t}{\sigma_w^t} \tag{3.7}$$

where μ_w^t and σ_w^t are the time-aware mean and standard deviation of word w, respectively (see Section 3.5.2).

Definition 3.9 (Bursty word) *A word w is considered bursty if the burstiness degree of w is at least two standard deviation above the mean, i.e., $burst(w,t) \geq 2$.*

Since the $\mu_w^t \pm 2\sigma_w^t$ region contains 95% of the data under the assumption of a normal distribution, the probability that s_w^t is generated from the normal distribution when $burst(w,t) \geq 2$ is $\frac{1-0.95}{2} = 0.025\%$ and therefore w can be identified as bursty.

If w is a new word that has been not observed in history, i.e., having $\mu_w^t = 0$ and $\sigma_w^t = 0$, we assume a prior noise level of $\sigma_w^t = s_w^t/f$. That is, the initial noise level is $1/f$-th of the observed word signal. In our work, we set $f = \log_2(|\mathcal{W}_t^w| + 1)$, so that, and according to Definition. 3.9, word w should be observed at least 3 times at snapshot t in order to treat it as bursty word. This restriction on the number of occurrences of a new word is useful to mitigate the impact of the out-of-vocabulary words appearing frequently in social media.

However, social media is very noisy with many words appearing suddenly and diminishing directly. Moreover, event-related words are likely to be used before or/and after their respective events. For example, a group of friends might talk about a soccer match that they plan to attend a few hours before its actual start time. This leads to a number of false positives (words that are wrongly identified as bursty), which degrades the accuracy of the extraction procedure. We call the keywords mentioned outside their event period *temporal outliers*. A main factor that helps in distinguishing

actual keywords from temporal outliers is that the later is usually less frequently recurring at consecutive snapshots. That is, actual keywords are more likely to recur in temporally-close snapshots. In the following, we introduce another characteristic of words, which helps mitigate the influence of these outliers.

3.6.2 Word Recurrence

In this section, we study the recurrence characteristic of a word published at the current snapshot and provide a quantitative measure describing its repetitive occurrences at close snapshots. Words recurring more frequently during $\mathcal{W}$ than during history should be identified. As discussed in Section 3.5.2, the inter-arrivals are assumed to be exponentially distributed, then the probability density function that models these inter-arrivals is

$$f(a_w^t; \lambda_w^t) = \lambda_w^t \times e^{-\lambda_w^t a_w^t} \tag{3.8}$$

where λ_w^t is the expected inter-arrival rate, as described in Section 3.5.2. To quantify and normalize the degree of recurrence of a certain word at snapshot t, we divide the likelihood of the observed inter-arrival time a_w^t by the maximum value of $f(a_w^t; \lambda_w^t)$, which occurs at the mode (0 in case of exponential distributions), i.e.,

$$\frac{f(a_w^t; \lambda_w^t)}{f(0; \lambda_w^t)} = e^{-\lambda_w^t a_w^t}. \tag{3.9}$$

We define the recurrence score of word w at snapshot t as

$$recurrence(w,t) := \begin{cases} 1 & \text{if } (a_w^t > 0) \text{ and } (\lambda_w^t = 0) \\ e^{-\lambda_w^t a_w^t} & \text{if } (a_w^t < \frac{\ln 2}{\lambda_w^t}) \text{ and } (\lambda_w^t > 0) \\ 0 & otherwise \end{cases} \tag{3.10}$$

Case 1 indicates that if word w is not observed in history ($\lambda_w^t = 0$) and starts to occur during $\mathcal{W}$, then it is a new word, and thus, it acquires a weight of 1. In case 2, w is observed in history; however, it exhibits some recurrent pattern during $\mathcal{W}$ because its observed inter-arrival a_w^t is less than a specific cut-off threshold. We set this threshold to the median $\frac{\ln 2}{\lambda_w^t}$ of the underlying exponential distribution. The third case indicates that the word is not showing any significant recurrence pattern, and hence, is set to 0. Moreover, if w is observed once during $\mathcal{W}$, the recurrence score is set to 0.

Definition 3.10 (Recurrent word) *A word w is considered recurrent at snapshot t if the recurrence score of w is greater than 0, i.e., $recurrence(w,t) > 0$.*

This recurrence characteristic of a word will be used in the following section to filter out both temporal outliers and words that appear only once, such as typos.

3.6.3 Keyword Identification

In this section, we make use of both word characteristics, i.e., burstiness and recurrence, to identify keywords K^t at snapshot t and to assign a weight for each keyword (see Figure 3.8).

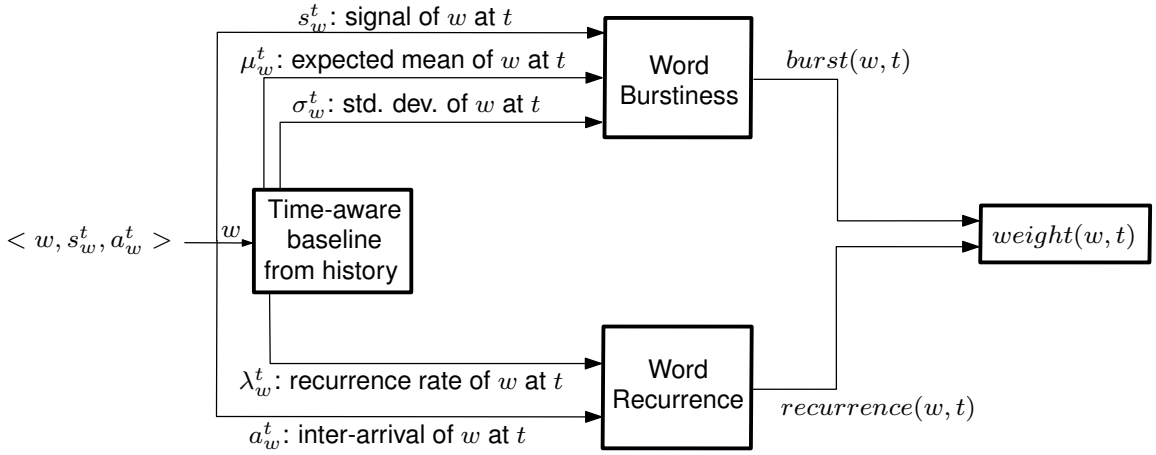


Figure 3.8: A block diagram for estimating a weight for word w.

We define a dynamic weight combining both characteristics as:

$$weight(w,t) := burst(w,t) \times recurrence(w,t), \tag{3.11}$$

which is the weight of word w at snapshot t. If w is related to a certain event, then its weight is likely to receive a high value during the event. Therefore, we can redefine the concept of a keyword as follows

Definition 3.11 (Keyword) *A word $w \in \mathcal{W}^t$ observed at snapshot t is a keyword if and only if w is both bursty and recurrent. Therefore, the set of keywords extracted from snapshot t is $K^t = \{w : weight(w,t) > 0 \wedge w \in \mathcal{W}^t\}$.*

As a result, a set of keywords K^t is extracted and a weight is estimated for each once a new snapshot elapses. At the next snapshot, the same steps are followed, and hence, some new keywords appear, others previously extracted may vanish or

have their weights changed. Some application examples on these keywords and their dynamic weights are discussed in Section 3.8.

3.7 Experiments

In this section, after describing the datasets, we evaluate different aspects of KeyPicker. More precisely, we test its ability to extract event-related keywords (Section 3.7.2), to place such keywords among the top-weighted ones during the event period (Section 3.7.3) and to assign a dynamic weight evolving over time for each keyword (Section 3.7.4). Then, in Section 3.7.5, we compare KeyPicker against a baseline approach and a state-of-the-art framework. Finally, we evaluate the scalability of KeyPicker in processing a large stream of incoming microblogs.

3.7.1 Datasets and Infrastructure

To evaluate the performance of KeyPicker, we use three datasets of tweets retrieved using the public Twitter API[1]. In Table 3.2, a feature summary is given describing each dataset: the bounding box from which the tweets originated, the interval during which the tweets were collected, the history period used to estimate the usage baseline parameters, and the average number of tweets published per snapshot.

Table 3.2: The specifications of the datasets **MAD**, **UKR**, and **NYC**.

	MAD (Madrid)			**UKR** (Ukraine)			**NYC** (New York)		
		lon	lat		lon	lat		lon	lat
Bounding Box	from	-3.93	40.21	from	23.99	45.20	from	-74.63	40.50
	to	-3.33	40.64	to	40.51	53.26	to	-73.63	40.97
Time interval	from	2013/1/24		from	2012/6/22		from	2013/10/20	
	to	2013/2/08		to	2012/7/08		to	2013/11/30	
History	from	2013/1/24		from	2012/6/22		from	2013/10/20	
	to	2013/1/28		to	2012/6/29		to	2013/10/30	
# / **snapshot**	1772.4			648.5			6027.3		

Due to the unavailability of a ground truth for evaluation purposes, we manually checked several Web sources to retrieve some featured events (see Table 3.3) that occurred during a time period covered by the three datasets. These events are treated as study cases for evaluation purposes. Dataset **MAD** contains tweets originating from the city of Madrid in Spain and encompasses two main events: (1) The Classico

[1] `http://dev.twitter.com/pages/streaming\_api`, accessed Apr. 2012

match that started at 20:00[1] on Jan 30th, 2013 at Santiago Bernabeu stadium, and (2) the Fitur[2] event occurred during the time interval from Feb 30th, 2013 to Feb 3rd, 2013 at the trade fair center of Madrid (IFEMA)[3]. The second dataset **UKR** includes a number of UEFA 2012 matches with the final match between Spain and Italy being the most attractive event. The final match took place at 20:45 EET on July 1st, 2012 at the Olympic stadium. In dataset **NYC**, five featured events are chosen; two correspond to American football matches, one to the annual marathon in NYC, one to Macay's event, and the last to a musical concert. Recall that our main goal in this chapter is not to detect events, yet evaluating the ability of KeyPicker to extract related keywords, and to provide a dynamic weight for each keyword.

The length of the used snapshot and sliding window c were set to 1 hour and 6 hours, respectively. The experimental platform is based on an Intel Core i7 QuadCore (2.4 GHz) with 8GB memory on Ubuntu 12.04 (with Java 6 framework). To simulate the real-time processing of tweets, the tweets are chronologically-ordered, as described in Section 3.4, and the stream is replayed, one snapshot at a time.

3.7.2 Keyword Extraction

In this section, we test the ability of KeyPicker to extract keywords that are related to some real-world events and to filter out temporal outliers. Figure 3.9 shows the temporal profile of both the signal and baseline of word "morning" in dataset **NYC**. As can be seen, and due to using time-aware baselines, both estimates behave the same at different points in time. However, the word "morning", as well as other periodic words, will exhibit a bursty behavior every day at the same time, which increases the system's false positives.

Figure 3.10 shows the word signals, usage baselines, and weights of the words ("final", "giants", and "halamadrid"). On July 1st, 2013, the word "final" is identified as a keyword at a number of snapshots due to the occurrence of the final match in the UEFA 2012 champions league. This keyword is subject to suffering from temporal outliers because it can be mentioned in different contexts. In particular, for such a popular and attractive event, the keyword "final" tends to be mentioned even before/after the match starts/ends. However, and as can be seen in Figure 3.10b, KeyPicker can mitigate the impact of these outliers and assigns lower weights to them compared to the weights assigned to this keyword at snapshots close to

[1] All timestamps mentioned in this thesis are based on GMT time unless otherwise specified.
[2] International Tourism Trade Fair; a global meeting point for tourism professionals.
[3] the trade fair center of Madrid

Table 3.3: Featured events as case studies from dataset **MAD**, **UKR**, and **NYC**.

Dataset	Eid	Event Description	Keywords	Start time	Location
MAD	$E1$	The Calssico Match	halamadrid, classico, santiago, bernabu	2013/1/30 20:00	Santiago Bernabeu stadium
	$E2$	Fitur	fitur2013, IFEMA, stand	2013/1/30 11:00	IFEMA
UKR	$E3$	Final match (UEFA 2012)	final, spain, euro, uefa	2012/7/1 18:45	Olympic stadium, kiev
NYC	$E4$	Raiders v.s. Giants match	raiders, giants, rutherford, stadium	2013/11/10 18:00	MetLife Stadium
	$E5$	ING New York City Marathon	marathon, ing, nyc-marathon	2013/11/03	NYC
	$E6$	Saints vs. Jets	whodat, saints, saintsgame-day	2013/11/03	MetLife Stadium
	$E7$	Macy's Parade Balloon Inflation	macys, inflation, balloon, marching, parade	2013/11/27	NYC
	$E8$	Krewella with Gareth Emery	krewella, gareth, emery, pieroffear	2013/11/01	Pier 94 Manhattan
	$E9$	Argentina vs. Ecuador	argentina, equador, argenti-navsecuador, mundial	2013/11/15 22:00	MetLife Stadium
	$E10$	The Victoria's Secret Fashion Show	fashion, secret, victoria, vs-fashionshow, victoriasse-cret	2013/11/13	69th Regiment Armory

the match time. For example, on June 30th, a number of relatively high signals were observed for this word, as shown in Figure 3.10a. However, KeyPicker, by incorporating the *recurrence* characteristic of a keyword (see Section 3.6), assigns no positive weights to the keyword "final" up to 6 hours before the start time of the match. For keyword "giants", three peaks are identified, which pertain to three

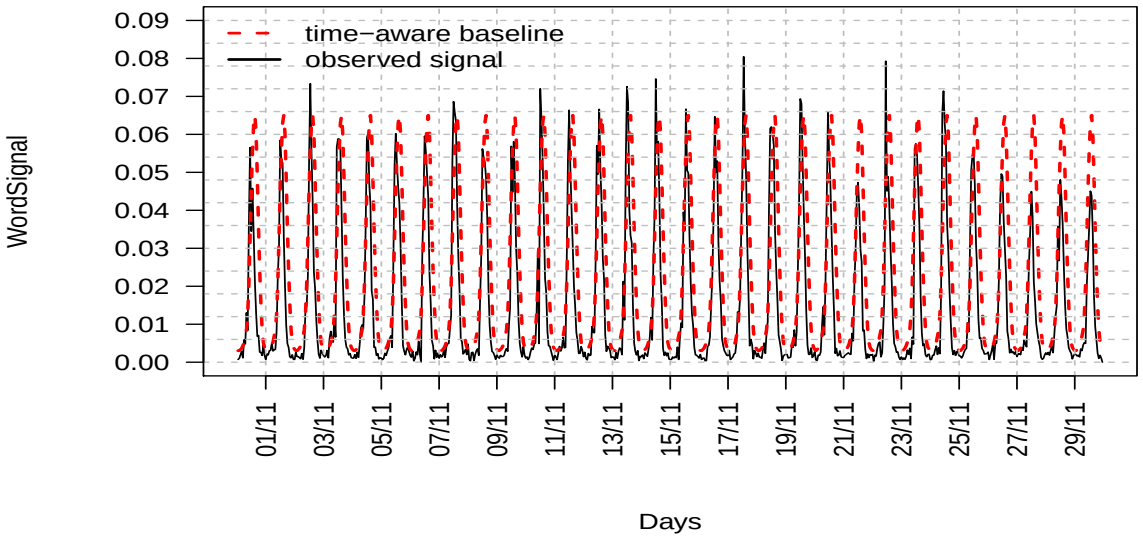


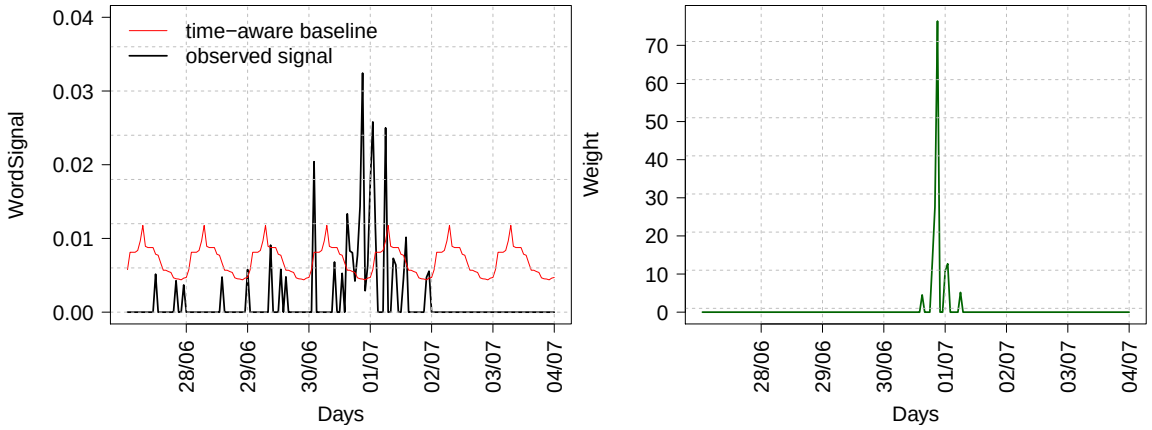
Figure 3.9: The dashed red curve depicts the time-aware baseline (mean) of the word "morning" from dataset **NYC**, while the other line corresponds to the actual word signal observed at each snapshot.

American football matches occurred during November 2013 in NYC (dataset **NYC**).

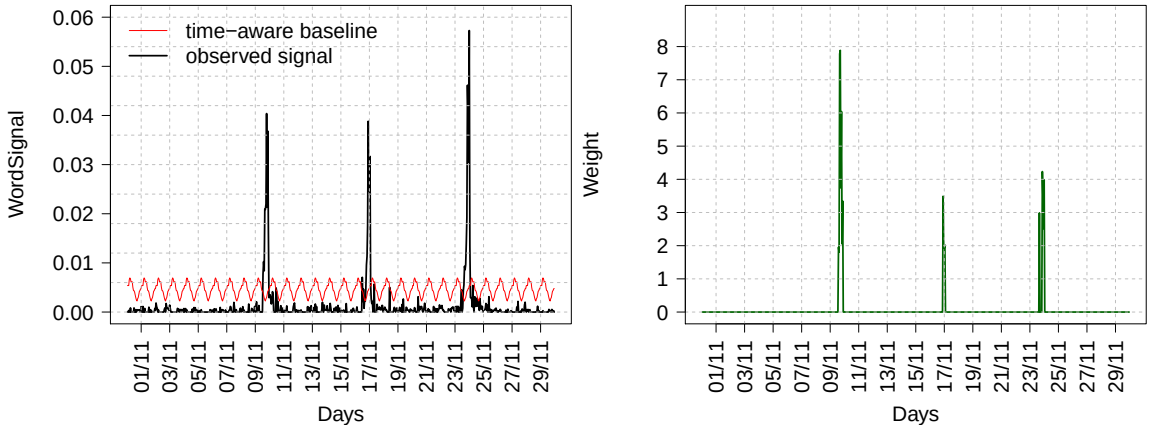
3.7.3 Top Keywords During Events

In the previous section, we evaluate our approach in assigning relatively high positive weights to keywords pertaining to some events. In this section, we conduct a closer analysis for the weights and investigate whether KeyPicker can place event-related keywords among the top-weighted ones extracted at each snapshot. For this purpose, and from the datasets **MAD**, **UKR**, and **NYC**, the top-10 keywords mentioned during a number of selected snapshots are listed in a descending order by their weights. From dataset **MAD**, as can be observed in Table 3.4, keywords related to $E1$ started to obtain significant weights that increase as the start time of $E1$ approaches. For instance, the word "clasico" received the weights (14.20, 23.25, 87.7) during the snapshots (18:00, 19:00, 20:00), respectively. The event $E1$ started at snapshot 20:00, which justifies the reason that the keyword "clasico" obtained the highest weight at that snapshot. Moreover, at snapshot 20:00, the majority of the top-10 keywords are Classico-related, e.g., "santiago", "bernabu", "clasico", and "halamadrid".

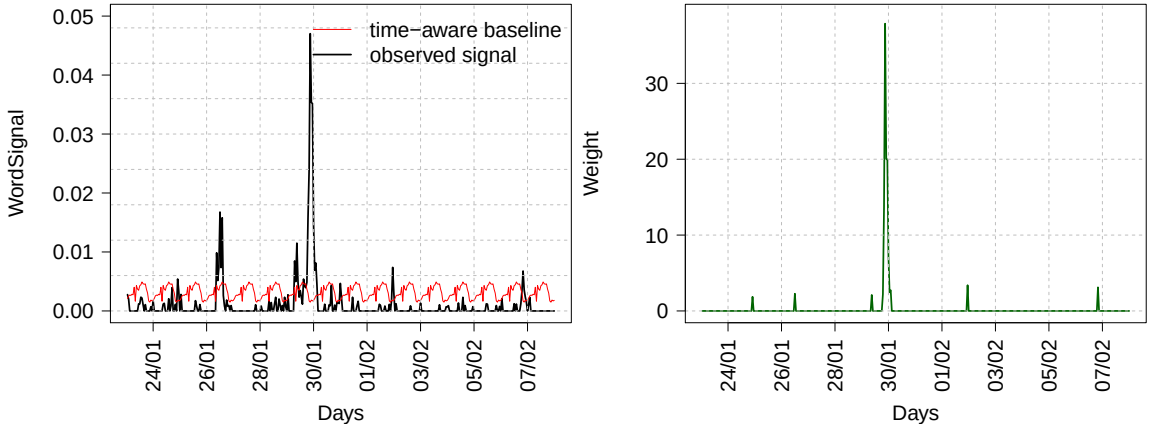
Similarly, keywords related to the events $E3$ and $E4$ exhibited a bursty and recurrent nature. Thus, these keywords were dominant with respect to the weights they obtained, as can be noticed in Table 3.4. For $E3$, which is the final match of the UEFA



(a) final (b) final



(c) giants (d) giants



(e) halamadrid (f) halamadrid

Figure 3.10: The temporal profile of three words (final, giants, and halamadrid) from the datasets **UKR**, **NYC** and **MAD**, respectively. Figures a,b, and c plot both the observed signals and the expected mean μ of the time-aware baseline of these words at a number of snapshots. Figures b, d, and f show the snapshots during which these words held positive weights, and thus, considered keywords.

2012 champions league, related keywords held high weights during multiple snapshots as shown in the middle part of the table. $E3$ stimulated people posting related tweets before, during, and after the match time interval. An important feature of KeyPicker that is noteworthy and apparent in this example is that keywords from different languages are extracted. This is because KeyPicker is language-independent and does not require defining a set of words a priori. For instance, some of the extracted keywords are Russian ("Олімпійський", "Испания", "Финал", "Смотрим" etc)[1].

3.7.4 Keyword Evolution

One interesting characteristic of KeyPicker to be evaluated is its ability to track the evolution of keywords over time, namely, its ability to assign dynamic weights for keywords, reflecting how the significance of each keyword changes as time progresses.

In Figure 3.11a, we consider event $E3$ and demonstrate the behavior of some keywords (final, spain, italy, nsc, uefa) observed in tweets from dataset **UKR**. The match started at 20:45pm EET, and as can be seen in the diagram, the keywords started to obtain significant weights at snapshot 19:00. Even after the match had ended, the weights remained relatively high in subsequent snapshots, which is a good indication of the important role that time-aware baselines play (see Section 3.5.3). For example, the word "spain" kept taking on high weights after the match ended, in spite of the drop in the number of users posting event-related tweets containing this word. Based on the time-aware baseline, temporally-consistent snapshots have even much smaller signals for "spain", which highlights its current observed signal.

Figure 3.11b shows the weights of keywords relating to the events $E1$ and $E2$ during the interval (05:00 2013/1/30 to 16:00 2013/1/31) from dataset **MAD**. Three groups (spikes) can be recognized in the figure. The first and third group refer to the same event "Fitur" with prominent keywords such as "feria", "fitur", and "ifema". The second spike corresponds to the Classico match. We can observe that related keywords, e.g., halamadrid, estadio, bernabu started to hold considerable weights at snapshot 19:00. Using a finer temporal granularity (one-minute snapshots), Figure 3.12 shows that the word "euro" was recognized as a keyword at several snapshots during July 1st, 2012.

[1] Олімпійський: Olympic, Испания: Spain, Финал: final, Смотрим: watching

Table 3.4: Top-10 extracted keywords relating to the events $E1$, $E3$, and $E4$.

snapshots from dataset MAD on Jan 30th, 2013 (Event $E1$)

16:00		17:00		18:00		19:00		20:00	
Word	Weight	Word	Weight	Word	Weight	Word	Weight	Word	Weight
fitur2013	56.43	anguila	38.90	motores	34.31	plataforma	28.12	clasico	87.76
stand	45.40	juas	23.08	ifema	22.87	contactar	25.04	santiago	61.43
loved	27.28	cubos	20.03	nao	20.59	clasico	23.25	bernabu	56.08
fitur	26.65	carajo	18.44	calentando	15.81	agotado	21.24	titularidad	54.26
sirven	13.88	fitur2013	9.97	clasico	14.20	vodafone	17.39	estadio	41.83
andaluca	10.88	pagamos	9.74	gregorio	10.63	bernabu	14.72	varane	41.47
amaneceres	10.73	caen	8.43	clsico	8.91	halamadrid	13.68	halamadrid	37.86
almeja	10.23	ministro	8.08	segovia	8.75	calentando	13.43	hala	30.76
calentar	8.46	orejas	7.87	aburres	8.64	asamblea	11.33	clasicazo	25.01
missed	8.25	dentista	7.86	fitur2013	6.75	ganara	10.63	realmadrid	22.95

snapshots from dataset UKR on July 1st, 2012 (Event $E3$)

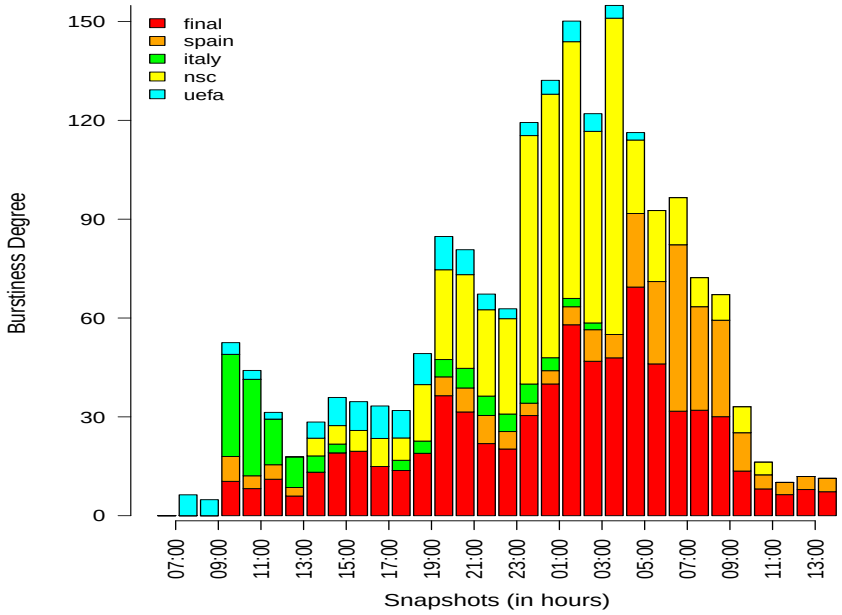
18:00		19:00		20:00		21:00		22:00	
Word	Weight	Word	Weight	Word	Weight	Word	Weight	Word	Weight
final	13.46	final	27.56	final	76.35	Испанцы	47.76	Испанию	117.7
stadium	12.45	olimpiyskiy	8.33	stadium	17.54	Болеем	9.60	болел	92.30
fanzone	10.74	italia	8.16	Финал	9.33	Чем	8.52	torres	18.70
olimpiyskiy	5.67	nsc	7.63	Испанию	7.43	Испания,	5.93	ппц	6.78
2012	3.83	Олімпійський	7.63	viva	7.30	Футбол	5.40	КАК	6.36
За	3.60	НСК	7.60	Смотрим	5.09	Италии	4.53	блядь	6.20
euro	3.22	2012	4.66	espaa	4.86	евро	4.39	Испания	3.39
nsc	2.91	лето	4.48	ukraina	4.52	espana	3.93	Финал	3.32
Олімпійський	2.80	Испания	4.38	2012	4.47	Испания	3.81	жалко	2.86
НСК	2.39	Последний	4.14	euro2012	3.59	forzaitalia	3.56	viva	2.16

snapshots from dataset NYC on November 10th, 2013 (Event $E4$)

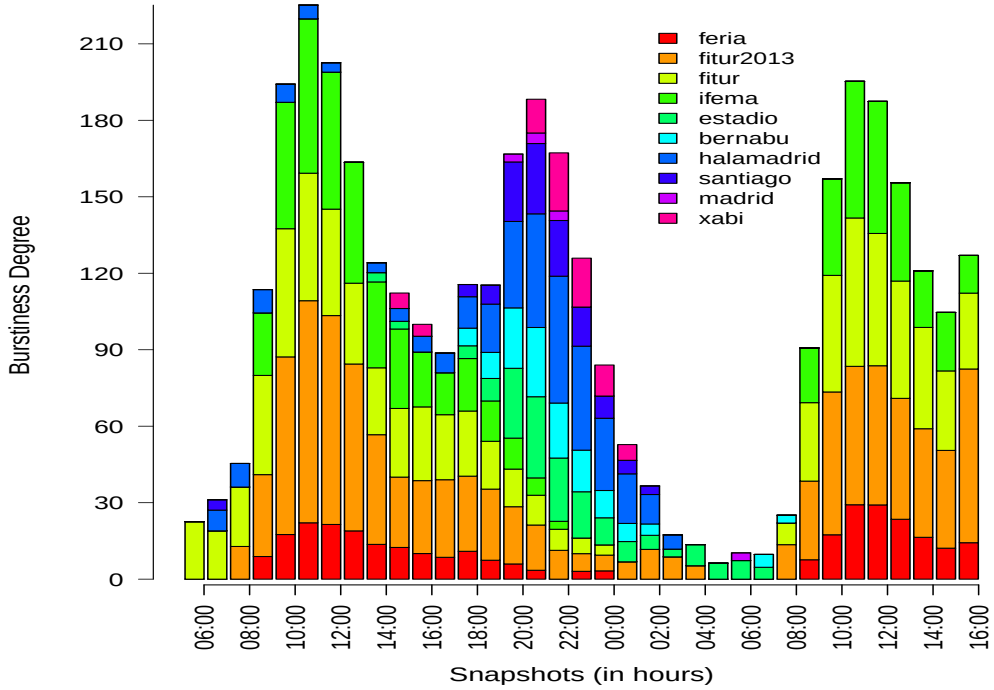
14:00		15:00		16:00		17:00		18:00	
Word	Weight	Word	Weight	Word	Weight	Word	Weight	Word	Weight
mongodb	22.32	raiders	18.38	raiders	45.34	raiders	44.02	engadget	80.82
thor	13.19	thor	7.86	spurs	35.39	engadget	26.96	raiders	56.90
reservoir	10.59	giants	7.78	thor	22.74	oakland	21.20	artrave	44.67
foster	10.21	options	6.17	pho	16.56	spurs	19.86	thor	26.11
philippines	8.49	tailgating	4.49	engadget	15.54	miamidol...	14.29	spurs	20.18
chantelle	7.17	gmen	4.23	cloisters	10.48	idolo	9.59	nygiants	14.78
bryant	5.91	sunderland	3.97	thegarden	9.89	nyknicks	9.38	izod	12.14
intoxicated	5.67	parking	3.60	nyknicks	8.17	woodson	8.79	oakland	11.58
tailgating	5.16	stadium	3.51	giants	7.88	thor	8.63	expand	9.54
movember	5.16	film	2.90	arsenal	7.32	channing	8.41	bigblue	7.24

3.7.5 Qualitative Evaluation

In this section, the aim is to compare the performance of KeyPicker against a baseline approach `BW` [3] and a state-of-the-art framework `BursT` [99] in extracting high-quality event-related keywords. The core point of this comparison is to determine the approach that can produce more informative keywords than the others. For the



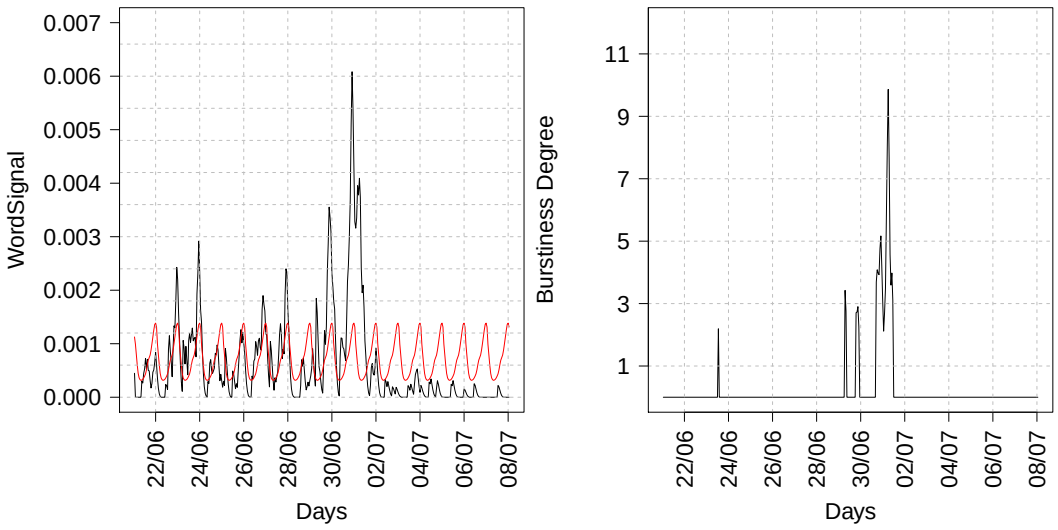
(a) The temporal distribution of keywords pertaining to the final match of UEFA 2012 from 07:00 2012/7/1 to 14:00 2012/7/2.



(b) The temporal distribution of a number of keywords pertaining to both the classico match and the Fitur event from 05:00 2013/1/30 to 16:00 2013/1/31.

Figure 3.11: Tracking the evolution of keywords' weights over time.

sake of evaluation, stop-words are not discarded in this experiment and used as a ground truth. That is, the approach avoiding to identify stop-words as keywords has a more informative nature than others. Figure 3.13a shows the percentage of stop-words



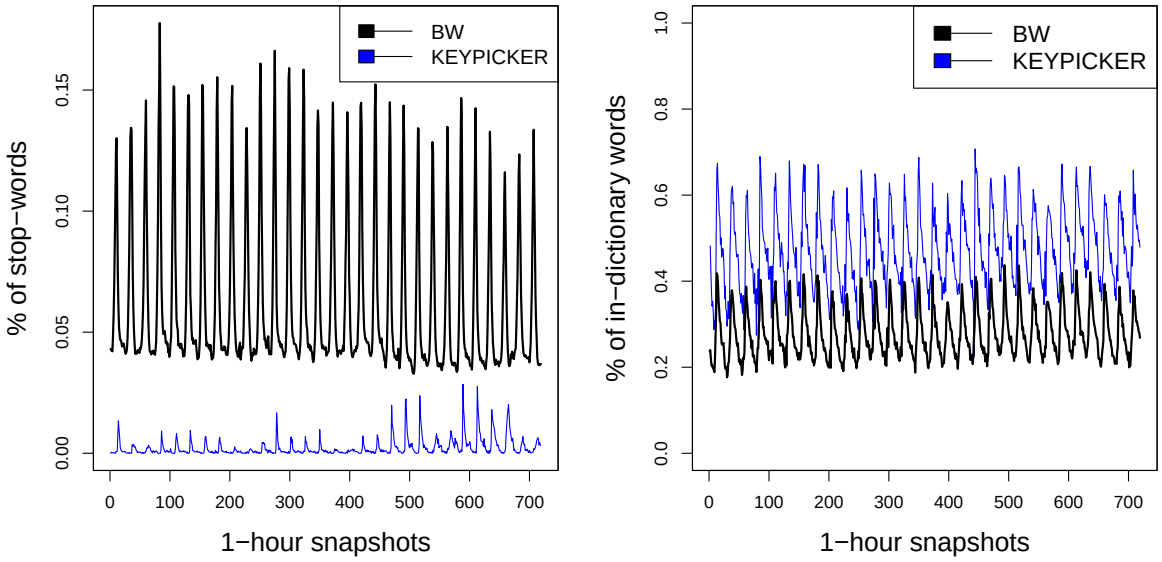
(a) (b)

Figure 3.12: Occurrences of keyword "euro" during (00:00 2012/06/22) to (23:59 2012/07/08). (a) The red curve shows the time-aware baseline (mean) of the word "euro", and the black curve corresponds to the actual word signal. (b) The weights of the word "euro".

at each snapshot during the period of study. The result is that KeyPicker considers much less keywords as stop-words than `BW`. This property is important not only for stop-words, but also for other words having a similar repetitive nature, e.g., "job" and "birthday". Similarly, KeyPicker outperforms `BW` in terms of the percentage of extracted keywords that exist in the WordNet[1] dictionary (see Figure 3.13b). The premise here is that the more keywords that can be found in the dictionary are extracted by a certain approach, the more reliable the approach is.

To conduct a more focused evaluation and to test how informative the keywords that KeyPicker extracts are, we investigate the events $E4$, $E5$, $E6$, $E7$ and $E8$ and report some statistics related to the top 100 keywords extracted using `BW`, `BursT`, and KeyPicker during the time period of each event. That is, the percentage of the extracted event-related keywords among the entire set of studied keywords (see Table 3.3) is computed for each approach. Figure 3.14 shows that KeyPicker outperforms the others in bringing the keywords of ongoing events as top keywords except for the event $E5$ that relates to NYC marathon. This is because this event is popular enough and there is a large number of users publishing marathon-related tweets. As a result, KeyPicker succeeds to perform well in the case of limited and unpopular events, e.g., event $E8$ that is related to a musical concert. This is

[1] `http://wordnet.princeton.edu`, accessed Dec. 2013



(a) (b)

Figure 3.13: Comparison between KeyPicker and BW in terms of (a) the percentage of stop-words identified as keywords, and (b) the percentage of keywords existing in the WordNet dictionary.

because KeyPicker considers not only the number of users mentioning a keyword but also how frequent this keyword is repeated at subsequent snapshots. Furthermore, KeyPicker determines whether a keyword is both bursty and recurrent, based on time-aware baseline parameters estimated from historical data. This is important to highlight the significance of a word even when it is mentioned by a small number of users.

A word cloud is plotted for the top-300 keywords extracted using BW, BursT, and KeyPicker for snapshot 2013/11/01 03:00. We chose this snapshot since it contains some tweets related to two events differing in terms of popularity, namely, Halloween and Krewella (see Table 3.3). From Figure 3.15a, we notice that KeyPicker is able to highlight keywords of unpopular events, where only a small number of related tweets are published. For example, the number of tweets related to event $E8$ is relatively small, and thus, corresponding keywords such as "krewella", "emery", "gareth", to name a few, are not that frequent. However, KeyPicker gives high weights to these keywords and puts them among the top keywords because it accounts for the contiguous recurrence of keywords (See Section 3.6). The approach of BW, as shown in Figure 3.15b, is sensitive to words appearing only once and diminishing in subsequent snapshots, such as "halloweeeeeeen" and "ahahahahahaha". BW considers them as keywords and assigns a positive weight to each. In Figure 3.15c, BursT gives relatively

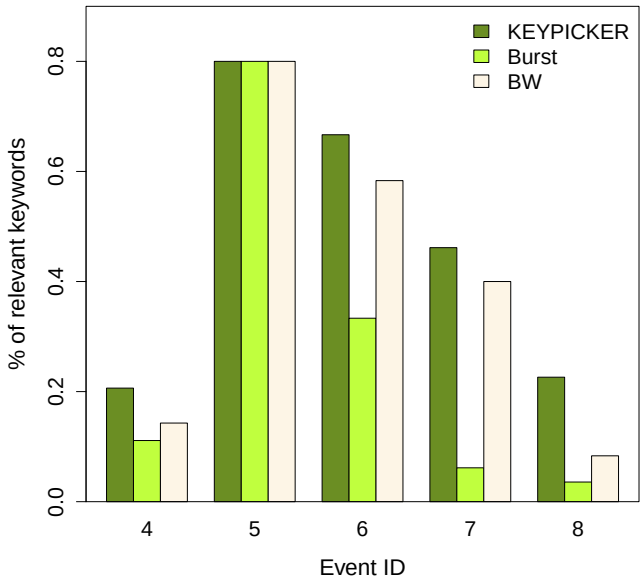
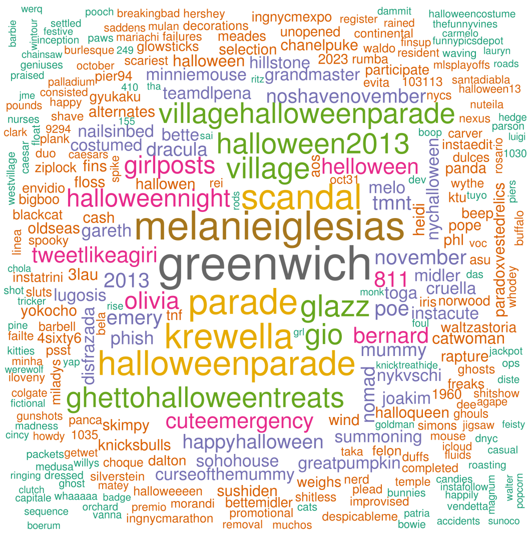


Figure 3.14: The percentage of event-related keywords among the top-100 keywords.

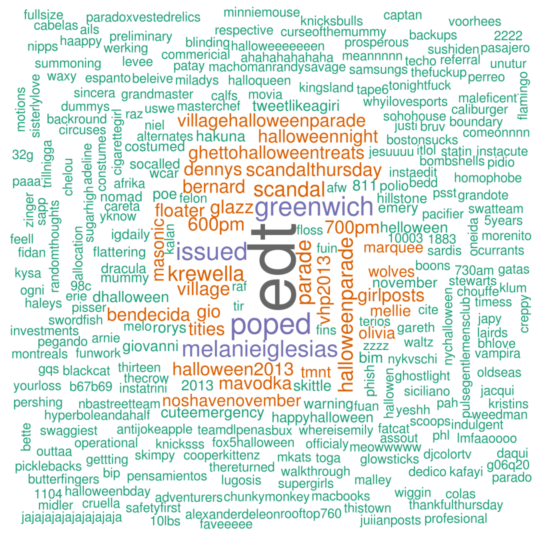
high weights to the words used on a daily basis, e.g., "wow", "happy", and "day". This, of course, results in producing too many uninformative keywords, which degrades the performance of applications utilizing such keywords, such as generating many non-event clusters when these keywords are used in an event detection framework.

3.7.6 Scalability

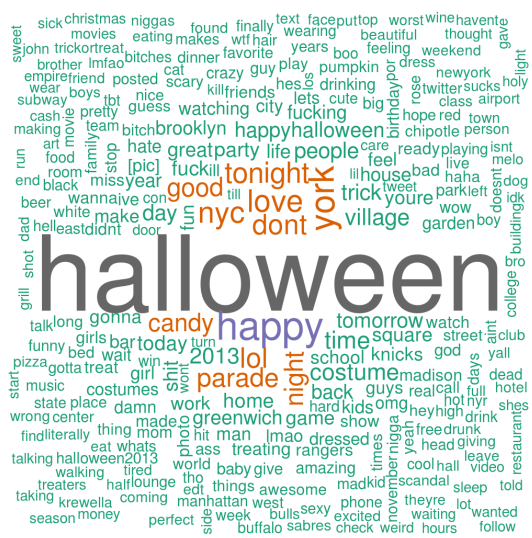
In this section, we evaluate KeyPicker with respect to its computational efficiency in processing a large-scale stream of microblogs. The goal is to show that our framework can finish the task of extracting keywords at a certain snapshot before the subsequent snapshot elapses. For this purpose, we chose the datasets **MAD** and **NYC** and computed the running time (in millisecond) that is required to process the tweets at each snapshot. In Figure 3.16a, the time needed to process the tweets in dataset **NYC** is reported for two tasks: content preprocessing and keyword extraction. The pairs (number of tweets, runtime) of each task and at each snapshot are aggregated, averaged, and ordered according to the number of tweets in each pair. The figure shows that the runtime of each task grows linearly in the number of tweets. Notice that the total time needed to process 11,000 tweets is about (20+60)=80 milliseconds, which indicates that KeyPicker can process a number of tweets that is about two times larger than the current actual Twitter stream rate (about 6,000 tweets/second). Figure 3.16b depicts the role of history materialization in improving the framework's efficiency. For example, only about 35 ms is required to process 4,000



(a)



(b)

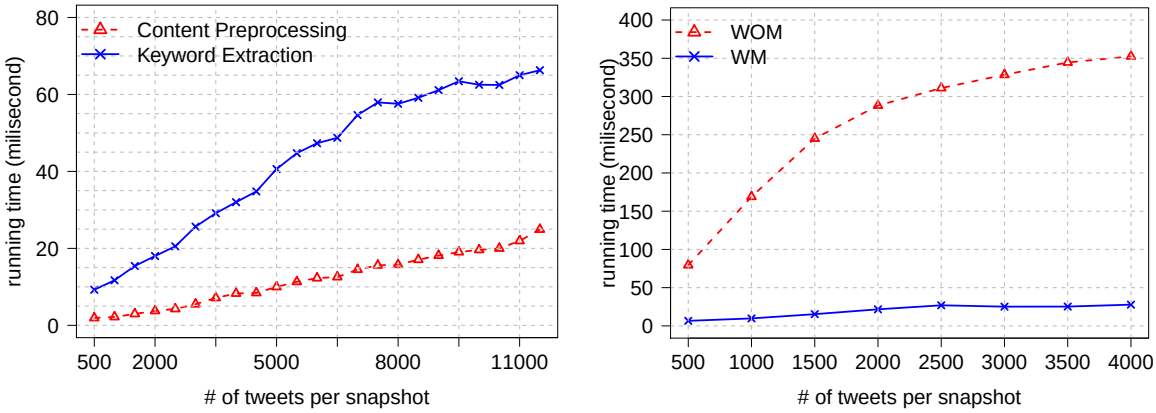


(c)

Figure 3.15: The word clouds in (a), (b), and (c) depict the top-300 keywords extracted using KeyPicker, `BW`, and `BursT`, respectively.

tweets from dataset **MAD** using history materialization (**WM**), while it is 350 ms without materialization (**WOM**).

On the other hand, it is obvious that KeyPicker can easily be parallelized. This can be achieved as follows. After the temporal index update task has been performed at snapshot t, the inverted index $\mathcal{W}^t$ is partitioned and each partition is assigned to a separate processing thread. Then, the extracted keywords from each thread will be combined to form K^t. For further performance enhancement, the word history can be



(a) NYC dataset (**NYC**) (b) Madrid dataset (**MAD**)

Figure 3.16: Computational efficiency of KeyPicker. (a) running time statistics for two tasks: content preprocessing and keyword extraction. (b) comparison between the required time to process tweets with (**WM**) and without (**WOM**) history materialization.

replicated and distributed among the processing threads to avoid the delay resulting from thread synchronization.

3.8 Summary and Discussion

In this section, we conclude this chapter by briefly summarizing the main contributions and outcomes of our keyword extraction framework and then present some applications.

3.8.1 Summary

In this chapter, we have introduced KeyPicker, a framework to extract keywords, i.e., words related to real-world events, from social media streams in a near real-time fashion. To capture this online settings, we exploit a sliding window on the streaming microblogs, so that each time the window slides, the most recent incoming microblogs are processed incrementally.

To judge whether a word can be considered a keyword or not, two word characteristics are described and quantified: word *burstines* and *recurrence*. A word is treated as a keyword if and only if it is bursty and recurrent. Combining these two characteristics into one estimate allows for handling the dynamics and noisiness of social media content and for filtering out temporal outliers (Section 3.6). These two characteristics are quantified based on the discrepancy paradigm, which measures the

distance between the current usage pattern of a word and an expected baseline of it. Such a baseline of each word is represented as a set of parameters that are estimated from historical observations. To produce trustworthy word characteristics, these parameters need to obtain up-to-date and reliable estimates. For this, our contributions regarding managing and estimating the baseline parameters from history are summarized as follows: 1) history update: including new words, e.g., new hashtags in Twitter, and excluding diminished ones. This is important to capture the dynamics of social media content. 2) Time-aware baseline estimation: coping with temporal heterogeneity that occurs due to the fluctuation of word usage at different snapshots (Section 3.5.3) by conducting a periodic history division. 3) history materialization: maintaining only the statistics that are sufficient to estimate the baseline parameters and are helpful in updating the history efficiently.

The experimental results show the performance of KeyPicker in incrementally extracting keywords from social media streams and assigning a dynamic weight for each keyword. KeyPicker outperforms both `BW` [3] and `BursT` [99] as a result of its ability to handle temporal outliers and to estimate baseline parameters from a separate history period not overlapping the sliding window, which is important in breaking the correlation between contiguous snapshots.

More on temporal heterogeneity. For simplicity, and based on domain knowledge, we divide the history of each word into periodic intervals of length one day, assuming that the majority of words have usage patterns that recur daily. However, some words are periodic, but recur differently, e.g., weekly or monthly. Hence, using the daily periodic intervals for such words to estimate their baseline parameters leads to identifying these words as keywords each time they appear, which increases the false positives and declines the accuracy of KeyPicker. This shortcoming of KeyPicker can be tackled by accomplishing a dynamic division of word history, which can be realized as a module to be plugged into KeyPicker.

3.8.2 Applications of Extracted Keywords

The keywords extracted at each snapshot, along with their respective weights, can be used as building blocks for various useful applications. Apart from the traditional trending topics identification, one can think of other types of applications such as:

Event-related microblog identification. The majority of content published via social media sites is noisy and contain no useful information about real-world

events. Thus, excluding such content is a key issue and an important preprocessing step to data mining in social media. In the context of event detection, identifying event-related microblogs among the noisy ones will lead not only to extract reliable event information, but also to improve the computational efficiency. This is because the majority of microblogs will be disregarded at an early stage. The principle here is that the more keywords with high weights a microblog has, the more likely the microblog pertains to a certain ongoing event. This particular application of extracted keywords is discussed in more detail in Chapter 4.

Localized events detection and tracking. The keywords extracted from social media might belong to one or more events occurring at different locations. On the other hand, the steadily increasing adoption of GPS-enabled mobile devices has recently increased the percentage of geo-tagged microblogs, which in turn, has allowed for the detection of another form of events, i.e., *localized events*. A localized event is an event that occurs at a certain location, e.g., musical concerts. In the next chapter, we will investigate the spatial dimension of each extracted keyword, handling related spatial problems and identifying those keywords with limited spatial focus, i.e., *local keywords*. In Chapter 4, as an application, we exploit the extracted keywords, along with their respective weights and spatial focuses to generate potential localized events and to track their evolution over time using the dynamic weights associated with each keyword.

Chapter 4

From Local Keywords to Localized Events – Detection and Tracking

In the previous chapter, we have studied the temporal characteristics of words published via social media sites. As a result, we are able to provide a reduced space of event-related keywords by filtering out noisy words. This process is performed in an online fashion to deal with the immense amount of incoming messages. In this chapter, we focus on the spatial characteristics of such informative keywords towards the main goal of detecting and tracking real-world events. These events, referred to as *localized events*, take place within a specific geographic region and last for a certain period of time, e.g., musical concerts or soccer matches.

4.1 Introduction

The timely detection and analysis of events from social media has been and is an active research area as it can provide information about real-world events faster than via news agencies. For example, Chunara et al. [50] proved that the trends of Cholera epidemic occurred in Haiti in 2010 were observed in Twitter messages and correlated in time with official sources, yet they were available 2 weeks earlier. Furthermore, the introduction of GPS technology and the availability of GPS-enabled mobile devices contribute to increasing the number of geo-tagged microblogs, i.e., microblogs enriched with location meta-data, in particular, geo-coordinates. As a result, it becomes possible to start studying and detecting localized events that occur at specific locations. The real-time detection and analysis of such events raises the level of the situational awareness, especially in disaster scenarios, and supports building location-aware recommendation systems.

However, an effective and efficient detection of localized events from social media faces a number of challenges: (1) The dynamics of social media content: microblogs have a high arrival rate, which requires a scalable system that can digest a large number of incoming microblogs and promptly update the state of previously detected events. (2) Spatial outliers: an event-related keyword might show a non-local behavior when people mention it far away from the event location. A mechanism is needed to regularize the spatial distribution for such a keyword in order to regain its actual local characteristics. (3) Spatial sparsity: relying only on geo-tagged microblogs results in a sparse distribution that does not reflect the actual geographic extent of a keyword, especially when using a fine-grained spatial resolution. (4) Studying the spatial distribution of each single keyword over space using a fine granularity is in general a computationally expensive process, especially when accumulating statistics from microblogs published during a certain time window.

Motivated by these aspects, we present LocEvent, a framework in support of detecting and tracking localized events in an online fashion. As can be seen in Figure 4.1, LocEvent depends on the well-known feature-pivot paradigm (see Chapter 2) that is based on extracting features (event-related keywords) and then clustering them to form potential events. To ensure an online processing of incoming microblogs, the sliding window model described in Chapter 3 is used here as well. Each time a new snapshots t elapses, LocEvent performs the following 4-stage procedure:

(1) Keyword extraction: A set of keywords are extracted using the approach described in Chapter 3.

(2) Focus of local keywords. In this stage, the keywords having a spatially limited extent are identified and their central locations, called *spatial focus*, are estimated. For this, the following operations are performed: (a) The microblogs containing at least one keyword are inserted into a spatial index to speed up the computation. (b) The spatial signature (distribution) of each keyword is estimated and adjusted to handle spatial outliers and sparsity. (c) Keywords having local geographic scope are identified. (d) The spatial focus of each keyword is estimated using a fine-grained spatial resolution.

(3) Event cluster generation. The keywords pertaining to localized events are then incrementally clustered based on their spatial focus using a density-based clustering algorithm.

(4) Cluster scoring. Each cluster is scored using some features describing its structure. A cluster's score reflects its evolution and significance over time.

LocEvent follows this 4-stage procedure at the next snapshot to incorporate new local keywords in the clustering, and so forth. Updating the spatial index requires adding content from the recent snapshot and eliminating the content of the expired snapshots. The contributions of LocEvent are summarized as follows:

(1) LocEvent handles both spatial sparsity and outliers. As for handling spatial outliers, two novel spatial regularization techniques are proposed: graph- and gazetteer-based (Section 4.5.1) regularization. Then, we use a non-parametric kernel density estimation to cope with spacial sparsity. Our work published in [3] builds the basis of these regularization techniques.

(2) A hierarchical space-partitioning index structure is employed to ensure a fast pruning of the geographic space while checking the locality of keywords and estimating their spatial focus. Our basic approach for local keyword identification was published in [2]. However, in this chapter, we build upon this approach and utilize this index structure to boost scalability.

(3) LocEvent tracks down the evolution of each detected event until it diminishes by assigning a dynamic score to each cluster. This score is also important in describing the relative significance of its respective cluster and in distinguishing event clusters from noisy ones.

The chapter is organized as follows. In Section 4.2, we discuss research efforts aiming at detecting real-world events from the content of social media. Related notations, concepts and the problem statement are considered in Section 4.3. In Sections 4.4, the spatial signature of keywords is defined and estimated. Then, the proposed approaches on adjusting the spatial signatures of keywords are presented in Section 4.5. The adjusted signatures are employed in Section 4.6 to identify local keywords and estimate their spatial focus. In Section 4.7, the steps required to spatially cluster local keywords and to score generated clusters are described. The experimental evaluation of our event detection framework is presented and discussed in Section 4.8. Finally, we conclude and discuss ongoing work in Section 4.9.

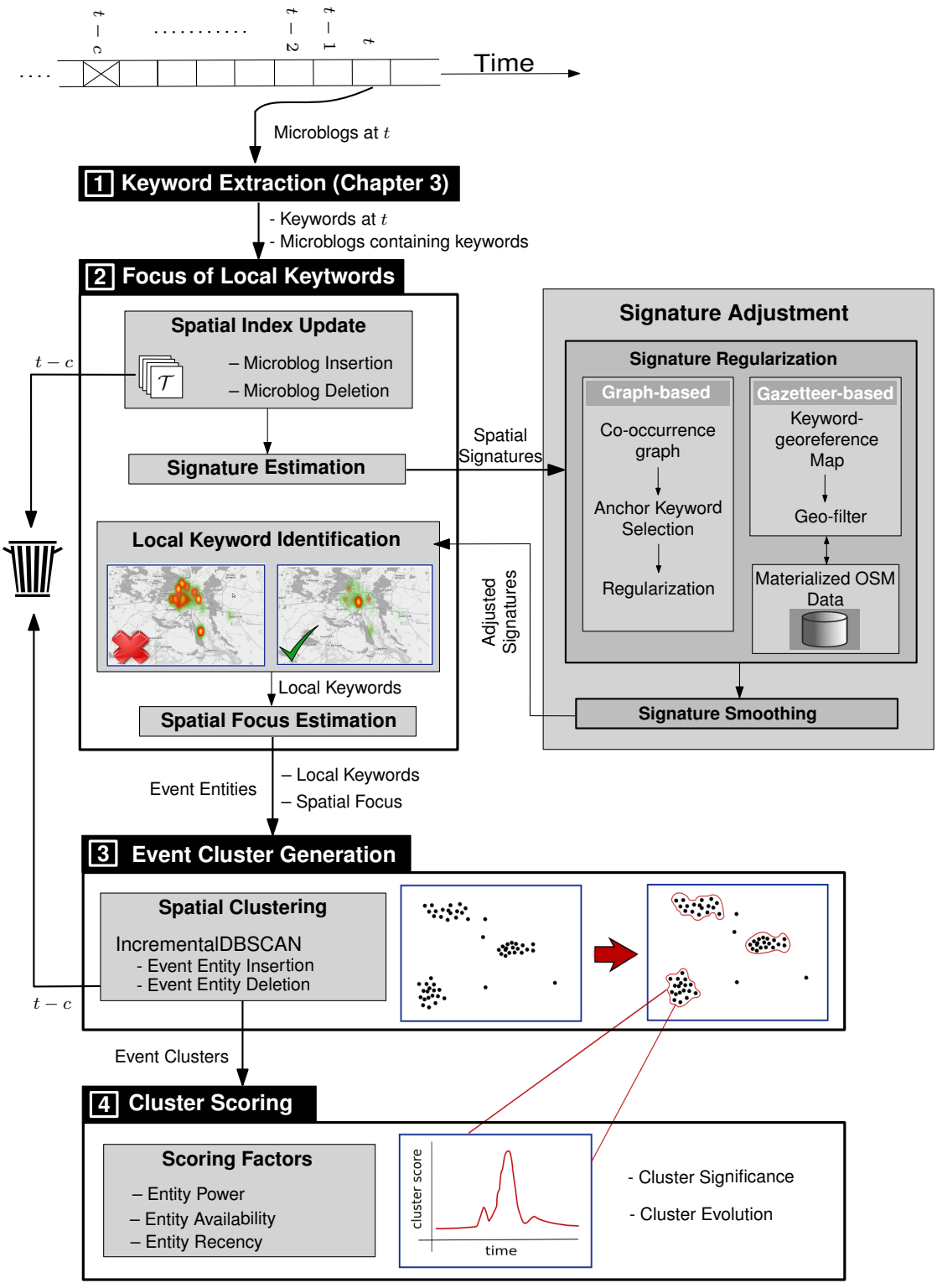


Figure 4.1: System overview of LocEvent. After extracting keywords in Stage-1, the spatial distribution of each keyword is estimated and spatial problems are handled. In Stage-2, the spatial focus of local keywords is estimated and used to spatially cluster such keywords in Stage-3. The generated event clusters are scored in Stage-4.

4.2 Related Work

A surprisingly increasing interest in detecting and tracking real-world events from social media has recently been witnessed. A large number of related research papers have been published in this context. In Chapter 2, a number of these event detection methodologies have been studied and categorized along three dimensions: (1) Event-entity type (document-pivot versus feature-pivot); (2) Detection task (NED versus RED techniques); and (3) Learning task (supervised versus unsupervised learning).

To narrow down the space of related studies to those having problem settings and objectives similar to LocEvent, we focus on the approaches that are adapted for social media content, are feature-pivot methods, and run in an online fashion. First, we discuss the efforts trying to extract event information by considering only the temporal dimension. Then, the studies that account for both spatial and temporal dimensions of messages are addressed.

Events over time. Li et al. [103] introduced *Twevent*, a framework to identify bursty word segments from tweets and cluster them using a k-nearest neighbor graph. They exploit Wikipedia to filter out non-event clusters. The approach by Parker et al. [122] tracks public health trends from medical tweets. Wikipedia is also utilized to associate the trending word sets with medical topics and screen out non-medical trends. Finally, medical trends are aggregated to detect shifts in public health conditions. Aggarwal and Subbian [8] utilize both the social links and the content of microblogs to improve upon the content-only similarity metric. Furthermore, they used a sketch-based technique to efficiently compute the structural similarity between clusters. Cataldi et al. [41] model the life cycle of terms using a novel aging theory and connect related emerging keywords after navigating a topic graph. For these approaches and similar contributions [15, 28], the spatial dimension is not considered, and hence, the resulting events are of global nature or large-scale local events.

Events over time and space. Sakaki et al. [130] built a supervised classification model to distinguish event from non-event tweets using a SVM classifier. The number of event tweets is modeled over time using a Poisson process to detect the occurrence of an event. Then, they apply Particle and Kalman filters to estimate the location of the detected event. To detect a new event, the classifier should be trained using a different set of event keywords, which is a limitation of their approach, standing against applying their approach to detect a broad range of events.

In the context of event detection from social media, *feature-pivot* detection is favored over document-pivot detection due to the noisy nature of social media and the short length of published messages. Feature-pivot detection is based on extracting event entities, e.g., bursty words, and then, clustering them to form potential events [46, 90]. Among a number of feature extraction techniques, those based on the discrepancy principle [57] have recently become popular by combining good scalability and effectiveness [3, 96]. The discrepancy paradigm measures the deviation between the current observation of a certain feature and its expected usage baseline.

Boettcher and Lee [35] proposed a framework that retrieves words W published during a time window from Twitter stream and builds subsets of length 1,2, and 3 words out of the power set of W. Then, the DBSCAN clustering algorithm is employed to find the subsets having limited spatial extent based on the geo-coordinates associated with the tweets of each potential event cluster. Finally, a classifier is trained to filter out non-event clusters. Watanabe et al. [154] estimate the location of non-geo-tagged tweets to increase the chance of finding localized events. Then, they search for place names and count the number of key terms that co-occur with each place name. However, their method fails to find the localized events when no place names pointing to the locations of these events are mentioned.

In summary, the approaches that are based on the feature-pivot paradigm and used to detect localized events from social media do not handle spatial outliers, i.e., when people mention event-related keywords far away from the locations of their respective events. This results in many non-event clusters and degrades the detection accuracy. Furthermore, to track the evolution of a newly-detected event cluster, one should keep track of its instances over consecutive snapshots using some similarity measures. However, LocEvent is proposed to cope with spatial outliers and to incrementally track the evolution of each detected event cluster over time. This is achieved using a 4-stage procedure as illustrated in Figure 4.1, where a spatial index structure is employed to expedite the detection process.

4.3 Preliminaries

In the following, we present the notations and concepts used in this chapter. Some of them are deeply discussed in Chapter 3, in particular, those related to the temporal organization of microblogs. For the sake of completion, we briefly revisit those required in this chapter. Then, we introduce the problem statement and the challenges to tackle.

4.3.1 Spatial Organization of Keywords

Each microblog $m = (S, id, uid, loc, time)$ contains a set of words S, a microblog identifier id, a user identifier uid, a creation time $time$, and geo-coordinates $loc = (lat, lon)$[1]. We assume that each geo-tagged microblog originates from within some geographic space G. The components (id, S, uid, and $time$) exist for each microblog, while only a small subset of the microblogs are geo-tagged, due to privacy reasons or the lack of underlying positioning infrastructure.

In order to apply a streaming algorithm based on a sliding window model, the timeline is split into fixed-length time intervals called snapshots $(\cdots, t-2, t-1, t)$, where t is the current snapshot. The last c snapshots correspond to the sliding window $\mathcal{W}$ holding the most recent microblogs. After snapshot t elapses, the published microblogs are processed using the methodology described in Chapter 3 to extract a set of event-related keywords $K^t \subseteq \mathcal{W}^t$ where $\mathcal{W}^t$ is the set of words published at t. In the previous chapter, we detailed our temporal index, where the words $\mathcal{W}^t$ are keys for an inverted index that allows for accessing the microblogs containing at least one keyword $k \in K^t$ in sub-linear time. That is, the set of microblogs published at snapshot t and containing keyword k is accessed using $\mathcal{W}_k^t$.

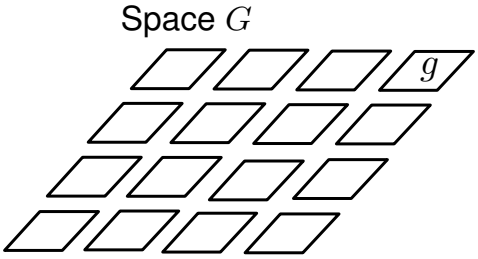


Figure 4.2: Partitioning space G using a regular grid.

As illustrated in Figure 4.2, the geographic area G of interest is viewed as a regular grid with a specific cell width. That is, the set $G = \{g_1, g_2, ..., g_{|G|}\}$ contains the cells composing the space G. Each microblog m is mapped to one of the cells based on its location $m.loc$. The cell width is treated as a configurable parameter that is chosen according to the level of granularity one needs to reach. We elaborate on choosing a suitable cell width for the adopted regular grid in Section 4.4.

[1]lat=latitude, lon=longitude

4.3.2 Localized Events

Localized events, such as festivals and musical concerts, are real-world events that take place at specific locations and last for a limited period of time. Such spatially limited events stimulates people to attend and/or participate. At the locations of these events, related microblogs, referred to as *social evidences*, are published by those people. Formally, we denote a localized event e as a tuple $e = (W^t, h^t, start, end)$ where W^t is the set of keywords describing event e, h^t is a convex hull representing the estimated location of e, *start* is the estimated start time of e, and *end* is the estimated end time of e. The first two parts, W^t and h^t, are time-dependent as our approach can capture the dynamics of localized events and shows how they evolve over time.

4.3.3 Problem Statement

Once the end point of snapshot t has been reached, there might be some localized events occurring somewhere in space G. From the microblogs published during the current snapshot t, we aim at *detecting the occurrence of new localized events or tracking the state of existing events*. For each detected event $e = (W^t, h^t, start, end)$, we want to *extract a list of ranked words (W^t) describing e well*. Moreover, we need to provide an estimation for *the location (h^t) of e, its start time (start), and end time (end) at a fine-grained spatio-temporal resolution*. For those events detected before snapshot t, we are interested in observing *how localized events evolve over time until they diminish*.

Keywords are the basic elements that make up localized events. For this, detailed analysis of these keywords is conducted with respect to their spread over space, identifying two spatial problems: spatial outliers and sparsity. Given the high rate of incoming microblogs, it is a challenging task *to detect and track localized events in an online setting taking into account such spatial problems.*

4.4 Spatial Distribution of Keywords

In this section, we first show how the spatial distribution of each extracted keyword $k \in K^t$ is computed. This spatial distribution is important to decide about the locality of keywords (Section 4.4.2). In Section 4.4.3, we address the spatial problems that contribute to distort the spatial distribution of extracted keywords, preventing proper decisions about the locality of keywords.

4.4.1 Spatial Signature

To describe the spatial spread of a keyword $k \in K^t$, we study the usage pattern of k at each cell $g \in G$. For this, we define the *spatial signal* of keyword k as follows.

Definition 4.1 (Spatial Signal) *The spatial signal of a keyword k observed at cell g during the sliding window $\mathcal{W}$ is the ratio between the number of users (N_k^g) publishing microblogs containing k and the number of all publishing users (N^g) at that cell, i.e.,*

$$\mathcal{S}_k^g := \frac{N_k^g}{N^g}. \tag{4.1}$$

Recall that in Chapter 3, we computed the signal for each word over the entire space G to identify keywords K^t. Here, we estimate the signals of only such keywords at a finer spatial resolution, namely at the cell level. Normalizing by the number of all publishing users N^g has a major impact on correcting the population bias. Without normalization, high-populated cells may lead to larger spatial signals even if the center of an event is located at another low-populated cell. To get a probability distribution for each keyword over cells, we estimate the density of a keyword k at a cell a by normalizing over its spatial signals in all cells:

$$\mathcal{S}_k^g = \frac{\mathcal{S}_k^g}{\sum_{i \in G} \mathcal{S}_k^i} \tag{4.2}$$

To refer to the spatial spread of a keyword k over the entire space G, we introduce the notion of *spatial signature*.

Definition 4.2 (Spatial Signature) *The spatial signature of a keyword k is the density distribution of the spatial signals of k in space G during $\mathcal{W}$, denoted $\mathcal{S}_k$. It is represented by the vector $\mathcal{S}_k := (\mathcal{S}_k^1, \mathcal{S}_k^2, ..., \mathcal{S}_k^j)^T$.*

The vector $\mathcal{S}_k$ retains only non-zero signals, which implies that $j \ll |G|$ as the distribution of microblogs over space is sparse, especially when using a fine spatial resolution.

4.4.2 Locality of Spatial Signatures

Recall that an important step in feature-pivot detection techniques is to identify event-related keywords. In the context of detecting localized events, such keywords are assumed to have a spatially limited extent and thus appear only in small parts of the geographic space. We call these keywords *local keywords*.

To test the locality of keyword k, the entropy measure [140] is used and applied to the spatial signatures $\mathcal{S}_k$ of k. It measures the minimum number of bits needed to represent the spatial spread of a keyword. If keyword k is equally distributed over $|G|$ cells, then the entropy is maximized and has the value of $\log_2(|G|)$. However, if it appears in exactly one cell, then the entropy will be 0. The entropy of the spatial signature $\mathcal{S}_k$ of keyword k over the entire space G is determined as follows:

$$H(\mathcal{S}_k) = -\sum_{g \in G} \mathcal{S}_k^g \times \log_2(\mathcal{S}_k^g). \tag{4.3}$$

Using this measure, a local keyword is defined as

Definition 4.3 (Local Keyword) *A keyword $k \in K^t$ is considered* local *if and only if $H(\mathcal{S}_k) \leq \phi$.*

The value of the threshold ϕ is chosen based on the locality level one needs to obtain. The smaller the threshold, the more strict decision is established. For example, when ϕ is set to 0, then only those keywords occurring at exactly one cell are considered local. Moreover, the resulting entropy value depends on the the cell width of the used grid. The cell width is chosen according to the level of granularity one needs to reach. For example, if the aim is to check the locality of keywords at the country level, a relatively large cell width, i.e., a cell width of 10km or more, is chosen. However, if one is interested in extracting local keywords related to localized events at the city level, a smaller cell width is considered, e.g., a cell width of 1km or less.

4.4.3 Spatial Problems

A manual inspection of the spatial signatures of some event-related keywords revealed undesirable distribution patterns that we refer to as *spatial problems*. Here, we present two types of such problems: spatial outliers and spatial sparsity.

Spatial Outliers

For some popular localized events, people who are not attending such events are stimulated to publish relevant microblogs, which implies that some related keywords will be mentioned far away from the event location. This type of keyword occurrences is called *spatial outliers*.

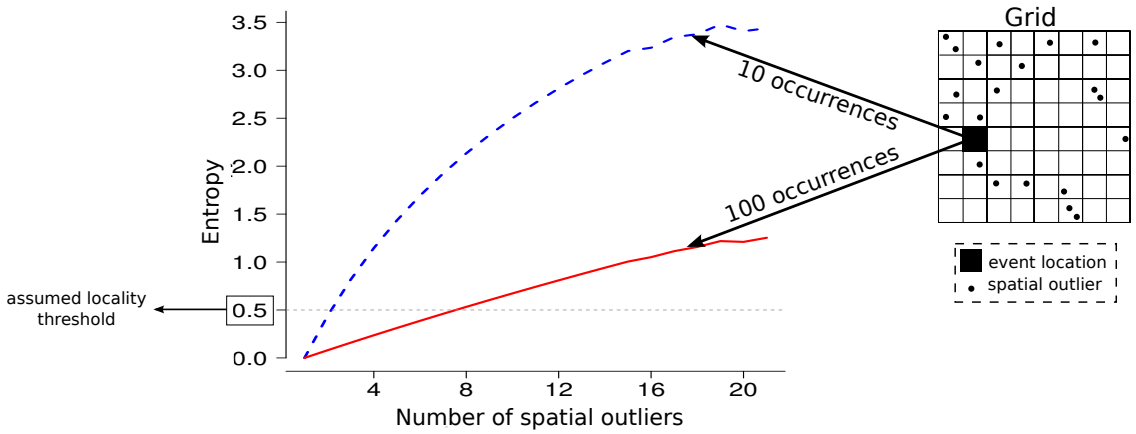


Figure 4.3: The impact of spatial outliers on the entropy of an event-related keyword. If this keyword is mentioned 10 times at the cell of the event, i.e., a low-populated cell, at least 3 spatial outliers make the entropy reach the locality threshold, identifying the keyword as non-local. While 8 outliers are needed to consider the keyword non-local in case of a high-populated cell that has 100 keyword occurrences.

Example 4.1 *Assume that the frequency of the keyword "final" at a certain snapshot at a cell containing a stadium is 100 while the total number of microblogs from that cell is 10,000. Then, the corresponding spatial signal will be* $100/10,000 = 0.01$. *Now, assume that there are 5 messages containing the keyword "final" out of 100 words observed at a low-populated cell far away from the stadium. This would give a spatial signal of* $5/100 = 0.05$, *which is larger than the spatial signal observed at the stadium.*

Spatial outliers contribute to the distortion of the spatial signature we wish to obtain for local keywords. Therefore, they will increase the entropy of the underlying distribution, leading to mistakenly considering local keywords as non-local. In addition, having a localized event taking place in a low-populated cell exaggerates this problem as the distribution in this case is more sensitive to outliers as they make the distribution approach faster to the uniform distribution as can inferred from Figure 4.3.

Spatial Sparsity

During a certain localized event, it is likely to notice some cells within the range of the event's location that contain no keywords describing the event. This is because the percentage of geo-tagged microblogs is in general low and not all cells are populated with users/microblogs. For instance, if a fire starts in a forest, there will be no microblogs posted from that place if there are no users. However, users from nearby cells could post such messages. We refer to this type of problem as *spatial sparsity*.

4.5 Spatial Signature Adjustment

In this section we present our solutions that adjust the spatial signatures of keywords and mitigate the adverse impact of spatial problems. In Section 4.5.1, two signature regularization techniques to handle spatial outliers are presented. We then detail a non-parametric signature smoothing technique to cope with the sparsity problem. As an application on this high-quality and adjusted signatures, we present in Section 4.5.3 a ranking methodology to distinguish social evidences among non-event microblogs.

4.5.1 Spatial Signature Regularization

Let us assume that keyword k that is related to a localized event occurring at cell g' suffers from spatial outliers. Our approach to handle this problem is to boost the density $\mathrm{p}_k^{g'}$ and lower the densities at other cells. As a result, this will decrease the entropy $H(\mathrm{p}_k)$, hoping that it reaches the desired locality threshold.

Graph-based Regularization

Given a set of keywords with spatial signatures affected by the spatial outliers, we aim at estimating regularized versions of such signatures. In other words, we want their spatial distributions reflect the spatial spread of potential events well.

Fortunately, and especially for localized events, one can find some keywords that are inherently localized and have a very limited spatial spread, i.e., they have a low entropy. For example, during a soccer match, there will be some keywords (e.g., the name of the stadium) that are rarely used far away from the stadium. The spatial signatures of such keywords are assumed to be stationary and perfectly represent the spatial scope of the event. We call such keywords *anchor keywords*. Through comparison of co-occurrence patterns of anchor keywords with those of non-anchor keywords, we regularize the spatial signatures of the latter accordingly and refer to them as *regularized spatial signatures*. Based on this rationale, the following general steps are used to regularize the spatial signatures of keywords:

(1) modeling keyword co-occurrences as a graph;

(2) choosing a subset of the keywords as *anchor* keywords that have spatial signatures of a high confidence, and

(3) traversing the graph's vertices and regularizing the spatial signatures of non-anchor keywords based on their connectivity to anchor keywords.

Co-occurrence Graph. First, a graph $G = (V, E)$ with a set of vertices V and a set of edges E is built to capture the co-occurrence of keywords $K^{\mathcal{W}}$ mentioned during the window $\mathcal{W}$. Each vertex $v_i \in V$ is mapped to exactly one keyword $k_i \in K^{\mathcal{W}}$, establishing a bijection between elements in V and $K^{\mathcal{W}}$. A weight is associated with each vertex v_i reflecting the ratio between the number of cells at which the keyword k_i occurs during $\mathcal{W}$ and the total number of cells. This measure is called *support*, defined as

$$supp(v_i) := \frac{\sum_{t \in \mathcal{W}} \sum_{g \in G} I(N_{v_i}^{t,g} > 0)}{c \times |G|}, \tag{4.4}$$

where $I(.)$ is the identity function and $N_{v_i}^{t,g}$ is the number of messages containing v_i and published from within cell g during t. An edge $(v_i, v_j) \in E$ between two vertices v_i, v_j is established if and only if the corresponding keywords co-occur at least once in cell $g \in G$ during $\mathcal{W}$. A weight is associated with each edge (v_i, v_j) indicating the percentage of cells at which the co-occurrence of the keywords k_i, k_j is observed during $\mathcal{W}$. Likewise, such a weight is defined using the concept of support:

$$supp(k_i, k_j) := \frac{\sum_{t \in \mathcal{W}} \sum_{g \in G} I(N_{v_i}^{t,g} > 0 \wedge N_{v_j}^{t,g} > 0)}{c \times |G|} \tag{4.5}$$

Anchor Keyword Selection. As mentioned above, spatial signatures affected by spatial outliers can be regularized by leveraging the spatial signatures of anchor keywords. The aim now is to identify such anchor keywords.

The main rationale behind identifying anchor keywords is that they potentially show high locality and geographic focus, in particular, when the anchor keywords relate to some localized events. Thus, these keywords tend to have more informative spatial signatures with low entropy values. We assume that anchor keywords are rarely used far away from the center of a localized event. Typical examples of anchor keywords are names of streets, cities, towns, landmarks etc. Let the set of anchor keywords extracted during snapshot t be denoted AK^t. We assume that non-anchor keywords are more affected by spatial outliers since they are more likely to be used far away from the location of an event. A simple step to extract AK^t is to choose those keywords that have spatial signatures with entropy values less than a given threshold, i.e., $AK^t = \{k | H(\mathcal{S}_k) < \phi \wedge k \in K^t\}$ In Section 4.8.3, we show how to choose a value for ϕ that yields the best results.

Signature Regularization. Given a co-occurrence graph and a subset of its vertices labeled as anchors from the above two steps. Algorithm 4.1 regularizes the spatial

signatures of non-anchor keywords in a given co-occurrence graph G based on their connectivity with anchor keywords. A set of regularized keywords RK is maintained to hold the keywords whose spatial signatures have been regularized. In Line 2, a set RK of anchor keywords is extracted using the approach discussed above based on the premise that such keywords have informative signatures. Thus, there is no need to regularize them. Then, the non-anchor keywords in V are traversed (Line 3) to regularize each of them.

Algorithm 4.1: Signature Regularization

Input: Co-occurrence graph $G = (V, E)$
Output: Set RK with regularized spatial signatures
`// Anchor keyword selection`
1 $AK^t \leftarrow getAnchors(V)$
2 $RK.add(AK^t)$
`// Traversing non-anchor keywords`
3 **foreach** $k \in (V - RK)$ **do**
4 $\quad RN \leftarrow getRegularizedNeighbors(k, RK)$
5 $\quad$ **if** $\neg RN.isEmpty()$ **then**
$\quad\quad$ `/* storing the keyword having the highest support with` k `*/`
6 $\quad\quad MC \leftarrow null$
7 $\quad\quad maxSupp \leftarrow -\infty$ `// the maximum support with` k
8 $\quad\quad$ **foreach** $ak \in RN$ **do**
9 $\quad\quad\quad \mathrm{p}(k, ak) = supp(k, ak)$
10 $\quad\quad\quad$ **if** $\mathrm{p}(k, ak) > maxSupp$ **then**
11 $\quad\quad\quad\quad MC \leftarrow ak$
12 $\quad\quad\quad\quad maxSupp \leftarrow \mathrm{p}(k, ak)$
13 $\quad\quad confidence \leftarrow \frac{supp(k,MC)}{supp(MC)}$
$\quad\quad$ `// Updating the signature of` k `using the signature of` MC
14 $\quad\quad \mathcal{S}_k \leftarrow (1 - confidence) \cdot \mathcal{S}_k + confidence \cdot \mathcal{S}_{MC}$
15 $\quad\quad RK.add(k)$
16 **return** RK

At each iteration, a set of regularized keywords RN that are neighbors of corresponding non-anchor keyword is returned (Line 4). If such a non-anchor keyword k is adjacent to at least one anchor keyword, then the anchor keyword MC that has the highest support is chosen to be the one with which the spatial signature of k is regularized (Line 8). In Line 13, the confidence that MC leads to k is computed by determining the conditional probability of the occurrence of keyword k given MC. If the non-anchor keyword k is observed at all cells where the anchor keyword MC is

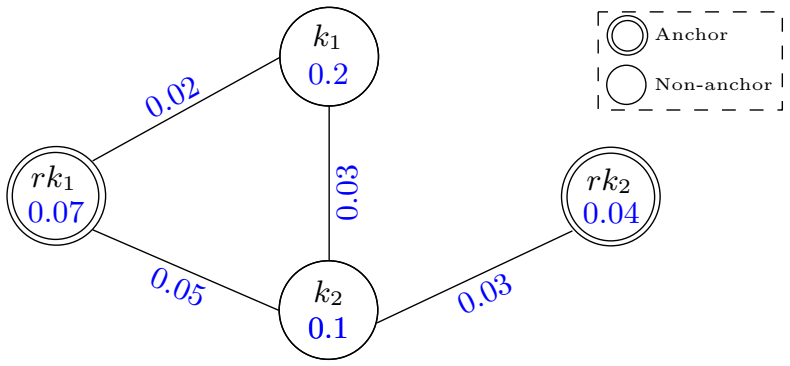


Figure 4.4: Example of a co-occurrence graph. The value in each vertex represents the support of the corresponding keyword. The value associated with each edge corresponds to the support of the vertices of that edge.

observed, then the confidence that MC leads to k is 1. This confidence metric takes on values in the range $[0, 1]$, and from the equation in Line 14, we notice that the larger the confidence value, the more the spatial signature $\mathcal{S}_k$ is affected by that of $\mathcal{S}_{MC}$.

Example 4.2 *Figure 4.4 shows a simple example of a co-occurrence graph with two anchor and two non-anchor keywords. The non-anchor keyword k_2 co-occurs with the two anchors rk_1, rk_2, and thus, one of them should be chosen to regularize the spatial signature of k_2. Since the value of $supp(k_2, rk_1)$ is greater than $supp(k_2, rk_2)$, the spatial signature of rk_1 is used to regularize that of k_2 with $confidence = \frac{supp(k_2,rk_1)}{supp(rk_1)} = \frac{5}{7}$. As a consequence, the regularized version of $\mathcal{S}_{k_2}$ is computed as follows:*

$$\mathcal{S}_{k_2} = \frac{2}{7} \cdot \mathcal{S}_{k_2} + \frac{5}{7} \cdot \mathcal{S}_{rk_1}.$$

If a non-anchor keyword is not pertaining to any localized event, we expect that the co-occurrence pattern of this keyword with anchor keywords will be minimal. Thus, it will not be highly affected by the regularization procedure. That is, non-anchor keywords of global events, e.g., *Mother Day*, preserve their broad spatial distribution. Hence, our signature regularization is effective in producing sound spatial signatures for keywords of both local and global events.

Gazetteer-based Regularization

We propose here another signature regularization technique that is based on utilizing the co-occurrence patterns between keywords and georeferences (place names) that are extracted by means of a gazetteer. The intuition is that if keyword $k \in K^t$ relates

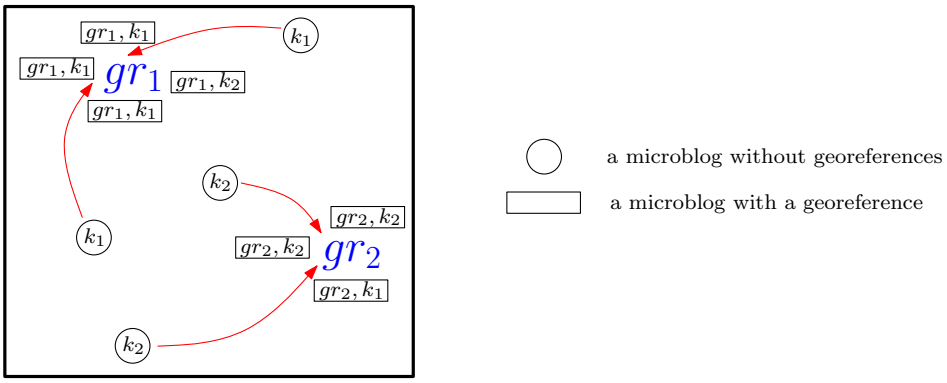


Figure 4.5: Keyword k_1 co-occurs more frequently with georeference gr_1 than with georeference gr_2, and thus, the signature of k_1 is affected more by gr_1. For the same reason, gr_2 has a larger impact on the spatial signature of k_2.

to a localized event, then k tends to co-occur more frequently with georeferences referring to the event location than with other georeferences. By this, it is possible to identify the cells that are candidates for the localized event described by keyword k and to determine how likely each candidate cell corresponds to that localized event. Figure 4.5 shows an example of how the location of two keywords k_1 and k_2 that co-occur with two georeferences gr_1 and gr_2 should be influenced.

We first build a *keyword-georeference* map that captures the co-occurrence patterns between keywords and georeferences, and then, a *geo-filter* is produced for each keyword in this map, which is used in regularizing its signature.

Keyword-Georeference Map. A georeference (also called toponym) is a phrase embedded in the content of a microblog and refers to a specific place on Earth. If a musical concert occurs at location gr during snapshot t, then there will be more microblogs containing keyword "music" together with gr than with other georeferences. This is because people tend to mention the location at which they observe or attend an event [2].

Suppose that the set of georeferences co-occurring with keyword k during t is denoted $\Omega_k = \{gr_1, gr_2, \cdots, gr_v\}$, where each $\Omega_k[gr_i] = (text, count, loc)$ is a tuple consisting of the textual phrase (place name), the number of times georeference gr_i co-occurs with keyword k, and the geo-coordinates $loc = (lat, lon)$ of that georeference, respectively. Hence, Ω refers to the set of keywords published during snapshot t and co-occurring with at least one georeference. The set Ω is called *keyword-georeference map*. Due to the sparse nature of this map, as only a subset of the extracted keywords

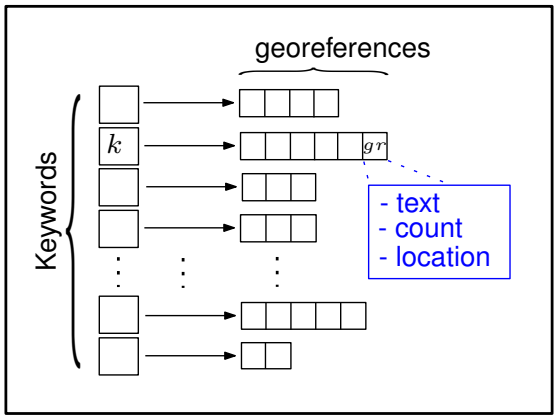


Figure 4.6: Keyword-Georeference Map. This map is realized as an inverted index where its keys are keywords and the postings of each key are the georeferences co-occurring with corresponding keyword.

map to one or more georeferences, it is implemented as an inverted index, as can be seen in Figure 4.6.

Algorithm 4.2 lists the steps needed to build the map Ω at snapshot t. In Line 2, the microblogs posted during t are processed to extract keyword-georeference pairs and to update map Ω accordingly. For each microblog, the embedded

Algorithm 4.2: Building Keyword-Georeference Map

Input: Keywords K^t, microblogs $\mathcal{W}^t$
Output: Ω map

```
1  Ω = {}
   // only considering geo-tagged microblogs
2  foreach m_i ∈ W^t ∧ loc_i ≠ {} do
       // extracting georeferences using OSM data source
3      gr ← extractGeoreference(S_i)
4      if (gr ≠ {}) ∧ (dist(gr.loc, loc_i) < ρ) then
5          foreach k ∈ (S_i − gr.text) ∧ k ∈ K^t do
6              if (k, gr) ∉ Ω then
7                  Ω.insert(k, gr)
8                  Ω_k[gr].count = 1
9              else
10                 Ω_k[gr].count += 1
```

georeference, if any, is extracted assuming that each microblog contains at most one georeference (Line 3). OpenStreetMap[1] (OSM for short) data is employed to

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

extract the georeferences. Despite that there are several Web services allowing for this type of georeference extraction, e.g., Open MapQuest[2], OSM data pertaining to space G is materialized and indexed to provide efficient and offline *geocoding*. Using this gazetteer-based georeference extraction procedure, we found that about 5.4% of the microblogs have georeferences. In addition, a geo-tagged microblog m_i contributes to building Ω only if m_i is published from within a circular buffer whose center and radius are $gr.location$, and ρ, respectively (Line 4). The parameter ρ is assigned a value of 500m to avoid place ambiguity, and the Haversive [141] formula is used to calculate the distance between the microblog and the georeference. Then, in Lines 5-10, each keyword k in microblog m_i (excluding those extracted as a georeference $gr.text$) is mapped to gr, and Ω is updated to include the (k, gr) pair.

Keyword Geo-filter. Now, we discuss how each georeference contributes to regularize the spatial signature $\mathcal{S}_k$ of keyword k over space G. First, for each keyword k, a probability distribution over the georeferences co-occurring with k is computed. For this, we utilize $\Omega_k[gr].count$, the number of times keyword k co-occurs with a certain georeference gr. We define the confidence α_k^g as the ratio between the number of times keyword k co-occurs with georeferences in cell g and the total number of times k co-occurs with georeferences over the entire space G at snapshot t. More formally, we have

$$\alpha_k^g := \mathrm{p}(g|k) = \frac{\left(\sum_{gr \in g} \Omega_k[gr].count\right) + 1}{\left(\sum_{g \in G} \sum_{gr \in g} \Omega_k[gr].count\right) + |G|} \tag{4.6}$$

where Laplacian Smoothing is used to mitigate sparsity and to eliminate zero probabilities. This confidence value reflects how likely the spatial signal $\mathcal{S}_k^g$ of keyword k at cell g will be affected in order to get the mode of the distribution close to the candidate event cell. As a consequence, if k belongs to a localized event and co-occurs with related georeferences, the entropy will decrease. The set of confidence measures, denoted α_k, of keyword k over G is referred to as the *geo-filter* of keyword k. After that, the spatial signals of k over G are regularized by multiplying each signal by its correspondence confidence in α_k and then by normalizing by the sum of the updated

[2] `http://open.mapquestapi.com/nominatim`, accessed June 2014

signals over the entire space, i.e.,

$$\mathcal{S}_k^g := \frac{\mathcal{S}_k^g \times \alpha_k^g}{\sum_{g' \in G} \mathcal{S}_k^{g'} \times \alpha_k^{g'}} \qquad \forall g \in G \tag{4.7}$$

To recap, in this section, we have presented two methodologies towards handling spatial outliers: graph-based and gazetteer-based regularization. In the following section, we will discuss how to handle the spatial sparsity problem.

4.5.2 Spatial Signature Smoothing

The spatial signal of a keyword referring to some localized event is expected to appear in a limited part of the geographic space. However, the spatial signatures encapsulating such signals are sparse, especially when a fine-grained spatial resolution is used, i.e., a regular grid with a large number of cells. This sparsity implies that a small number of cells might hold no signals while their neighbors do. In the following, we discuss how to smooth the spatial signals of a spatial signature by considering nearby cells, so that the influence of each signal is propagated to nearby cells.

In order to estimate a spatial signal of a keyword in a certain cell based on signals of surrounding cells, we employ a non-parametric kernel density estimation. Suppose that a keyword k is mentioned n times in space G. Each observation is represented as a 2D vector $\mathbf{x}$ including the location (longitude, latitude) of the cell at which k is observed. Hence, we have a bivariate random variable $\mathbf{X}$ from which the samples $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n$ (observations for keyword k) are drawn. The kernel density estimation then is

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} K_H(\mathbf{x} - \mathbf{x}_i) \tag{4.8}$$

where K_H is a bivariate density function that integrates to one. We use the common bivariate normal kernel defined as

$$K_H(\mathbf{x}) := \frac{1}{\sqrt{|H|}} K(H^{-1/2}\mathbf{x}), \tag{4.9}$$

where $K(\mathbf{x}) = \frac{1}{2\pi} \exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x})$. Matrix H is the covariance matrix that controls the amount and orientation of the smoothing process [153].

To use the regularized spatial signatures, we redefine Eq. 4.8 in terms of the spatial signature $\mathcal{S}_k$. For this, we group the observations into their corresponding cells and

sum over these cells instead of summing over the n observations. Thus,

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{|\mathcal{S}_k|} o_i \times K_H(\mathbf{x} - \mathbf{x}_i), \tag{4.10}$$

where o_i is the number of observations of keyword k at cell x_i. In fact, o_i is simply a proportion of the n samples determined by the density of the keyword k at cell i, i.e., $o_i = n \times \mathcal{S}_k^i$. Therefore, the kernel density estimation of keyword k at cell $\mathbf{x}$ is redefined as follows:

$$\hat{\mathcal{S}_k^{\mathbf{x}}} = \hat{f}(\mathbf{x}) = \sum_{i=1}^{|\mathcal{S}_k|} \mathcal{S}_k[i] \times K_H(\mathbf{x} - \mathbf{x}_i) \tag{4.11}$$

After applying the spatial signature regularization in Section 4.5.1 and signature smoothing as described in this section, an adjusted signature $\hat{\mathcal{S}}_k$ is estimated for each $k \in K^t$, capturing the sound spread of the keyword signals over space G after handling the spatial outliers and sparsity problems.

4.5.3 Social Evidence Identification

As an application, we design an online filtering system with the aim of identifying social evidences and filtering out noisy microblogs. At each new snapshot t, the system utilizes the extracted keywords, their spatial signatures and their weights discussed in Chapter 3, to accomplish this task. For this, we assign a score to each microblog in order to reflect its significance and evolution over time.

First, we estimate a score for each word in a microblog m separately. Then, an aggregate function is used to compute a score for the entire microblog. To estimate a score for each word $w \in m.S$, we consider two factors:

(1) the significance $sig(w, t)$ of its current weight $weight(w, t)$ among the values estimated during $\mathcal{W}$; and

(2) the current adjusted spatial signal $\hat{\mathcal{S}}_w^{m.loc}$ of that word at the location of the microblog $m.loc$.

Formally:

$$score(w) := \begin{cases} \delta \cdot sig(w,t) + (1-\delta) \cdot \hat{\mathcal{S}}_w^{m.loc} & \text{if } w \in K^t \\ 0 & otherwise \end{cases} \tag{4.12}$$

The value of $sig(w,t)$ for word w at snapshot t is estimated as a ratio showing how close the current weight is to the maximum one observed within $\mathcal{W}$, i.e,

$$sig(w,t) := \frac{weight(w,t)}{max\{weight(w,i)\}_{i\in\mathcal{W}}}. \tag{4.13}$$

The parameter $\delta \in [0,1]$ is for normalization and gives a weight for each factor. We choose $\delta = 0.5$ to distribute the influence of the two factors equally in the case of geo-tagged microblogs. However, for non-geo-tagged microblogs, we set $\delta > 0.5$ to give more weight to the significance factor as the location information is absent.

As can be inferred from Eq. 4.12, the value of $score(w)$ describes the spatio-temporal importance of word $w \in m.S$. In other words, if the word w appearing at the current snapshot t obtains the highest weight during $\mathcal{W}$, then $sig(w,c)$ has the value 1. At the same time, if the microblog m containing that word is posted from the center of a localized event, we will expect a high spatial signal value that approaches 1. Consequently, the overall score of word w will be close to 1 implying a considerable significance. Finally, a score for the entire microblog tw is computed by applying the mean as an aggregation function for all the scores $score(w)_{w\in m.S}$, i.e.,

$$score(m) := \frac{1}{|m.S|} \cdot \sum_{w\in m.S} score(w) \tag{4.14}$$

We expect that top-scored microblogs are of huge significance and that they hold useful information to analyze the triggering events. Different NLP techniques, e.g., text summarization and sentiment analysis, can be applied on these microblogs to extract further information about the events.

4.6 Spatial Focus of Local Keywords

After adjusting the spatial signatures of keywords as discussed in Section 4.5 and having more reliable spatial signatures for keywords, the next tasks to be performed towards detecting localized events are

(1) checking the locality of each keyword $k \in K^t$, and then,

(2) estimating the spatial focus (central location) of local keywords.

In Section 4.6.1, we define the concept of spatial focus and lists the main focus estimation steps. Then, in Section 4.6.2, we describe the spatial index employed to

accelerate the computation, and finally, we show how to exploit this index for locality check and focus estimation (Section 4.6.3).

4.6.1 Spatial Focus

Recall that one of the main objectives in this study is to estimate of the location of a detected event at a fine-grained resolution. This entails exploiting the location metadata associated with microblogs, namely, geo-coordinates, to find the spatial focus of local keywords.

Definition 4.4 (Spatial Focus $\mathcal{F}_k$) *The spatial focus $\mathcal{F}_k = (lon, lat)$ of keyword k is the geographic midpoint of a set of geo-coordinates associated with the geo-tagged microblogs that contain k.*

The geographic midpoint[1] is favored over the normal mean of points because it accounts for the spherical shape of Earth. Although the difference between the focus estimated using the geographic midpoint and that estimated using the normal mean can be neglected in the case of localized events, we choose to apply the geographic midpoint to generalize the detection approach. That is, when events of a larger spatial extent are to be detected, e.g., events at the country level, a grid of coarse-grained resolution (10km by 10km cells or even larger) is utilized. To compute the focus of a set of points using the geographic midpoint (see Section 4.6.2 for more details):

(1) the lon/lat coordinates of each point is converted into Cartesian (x, y, z) coordinates, then

(2) the mean of these coordinates is computed to get the midpoint in the Cartesian 3D space, and finally,

(3) this midpoint is converted back to lon/lat coordinates.

Checking the locality of each keyword $k \in K^t$ and computing the spatial focus of local ones is a computationally-expensive process, especially when using (1) a fine-grained spatial resolution and (2) a relatively large widow size. To cope with that, in the following section, we present a space-partitioning index structure where count and location information is maintained at multiple hierarchical levels, which provides efficient and incremental updates of such information.

[1] `http://www.geomidpoint.com/`, accessed June 2014

4.6.2 Hierarchal Space-Partitioning Index

In order to efficiently process microblogs with high-arrival rates, we employ a hierarchical space-partitioning index structure [134]. This type of spatial indexing recursively partitions the space into non-overlapping cells, starting at the root cell until a certain capacity threshold or a target resolution is reached. These partitions are organized in a hierarchical structure with the root cell referring to the entire space G and the last level referring to the finest (target) resolution as can be seen Figure 4.7. Quadtrees and pyramid index structures are typical examples of hierarchical indexing [147].

In this work, we adopt the pyramid index, because it allows for maintaining statistics (synopses), e.g., keyword counts, in internal nodes, which summarizes the information stored in the leaf nodes. Maintaining synopses in internal nodes, as will be discussed later, helps expedite the process of updating the index structure, deciding about the locality of keywords, and estimating their spatial focus.

Dynamic Pyramid Structure

The pyramid index structure is viewed as a tree $\mathcal{T}$ of height h. The nodes of the tree are divided into levels $(0, 1, \cdots, h-1)$ where level 0 and $h-1$ hold the root and the leaves, respectively. Each level corresponds to a regular grid partitioning the space into non-overlapping cells, such that each node in that level is mapped to exactly one geographic cell. The higher the level, the finer the spatial granularity used to partition space G. Let us denote the node j at level i in $\mathcal{T}$ as $\mathcal{T}_i^p[j]$, where p refers to the parent node at level $i-1$. The children of the node $\mathcal{T}_i[j]$ are denoted by $\mathcal{T}_{i+1}^j$ and represent the cells covered by the bounding box of the parent cell $\mathcal{T}_i[j]$. The root node $\mathcal{T}_0[0]$ corresponds to a bounding box covering the space G, and the leaves $\mathcal{T}_{h-1}$ reflect the finest spatial resolution to be targeted.

Maintaining synopses for the entire set of keywords published during $\mathcal{W}$ in internal cells of a pyramid requires extra space and additional overhead to update the index due to the high-arrival rate of microblogs. Therefore, we present a variation of the pyramid structure, called *dynamic pyramid* that has the following characteristics:

(1) A leaf node is created only if at least one keyword appears at the corresponding cell. In contrast to the partial pyramid [110], all leaves reside at the same level. Thus, it is a compromise between a partial pyramid and a complete pyramid [147].

(2) At level i, the number of partitions (cells) is at most l^{2i}. Our dynamic pyramid differs from other space-partitioning structures in its ability to assign a value larger than 2 to l resulting in a wider tree, and hence, a lower tree height when a specific spatial resolution is targeted.

(3) Not all data synopses maintained in leaf cells are summarized in parent cells. As we will show below, the used synopses are divided into two categories: (a) *keyword count*: it is used to estimate the spatial distribution of a keyword at each level in order to decide about its locality. This type of statistics is maintained in each cell of the pyramid. (b) *keyword location*: it is a summarized location information for a keyword and is used to estimate its spatial focus. These synopses are maintained only in the leaves of the pyramid.

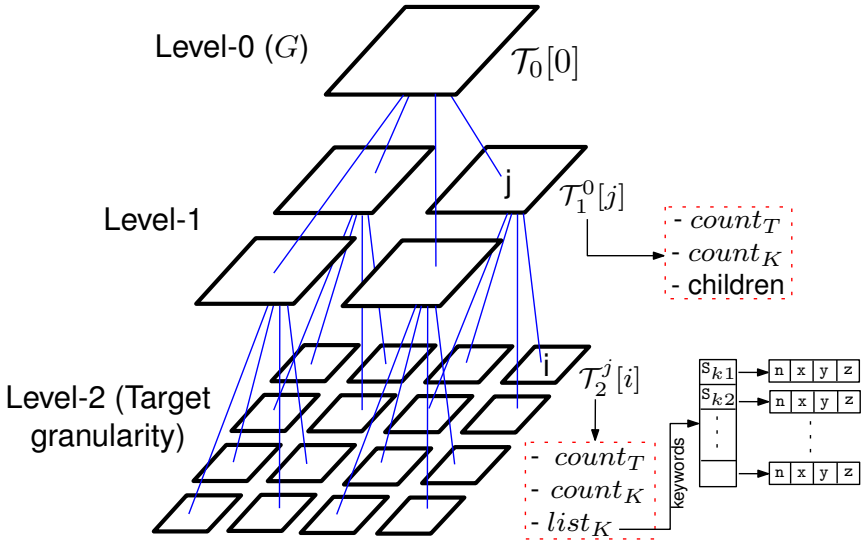


Figure 4.7: Dynamic pyramid to spatially index the microblogs published during $\mathcal{W}$. Internal cells hold count statistics required for locality check, while leaf cells contain, in addition to count statistics, location information needed in focus estimation.

As can be seen in Figure 4.7, each internal node $\mathcal{T}[j]$ in the tree contains:

(1) $count_T$: the number of microblogs published from $\mathcal{T}[j]$ during $\mathcal{W}$, which is used to both estimate the spatial signal at $\mathcal{T}[j]$ and accelerate updating the tree as discussed later.

(2) $count_K$: the number of microblogs containing a keyword and originating from within $\mathcal{T}[j]$ during $\mathcal{W}$. $count_K$ is maintained to check the locality of keywords. There is only one instance of $count_K$ at each cell, and therefore, once a new

keyword is examined, the value of $count_K$ is reset and initialized with respective count statistics accumulated from leaf cells.

Each leaf cell contains, in addition to the previous statistics, a list $list_K = (\mathtt{S}_{k_1}, \mathtt{S}_{k_2}, \cdots)$, containing synopses required to check the locality of keywords and compute their spatial focus. Each $\mathtt{S}_k = (\mathtt{n}, \mathtt{x}, \mathtt{y}, \mathtt{z})$ maintained in the leaf node $\mathcal{T}_{h-1}[i]$ has the following components: (a) $\mathtt{S}_k.\mathtt{n}$: the number of microblogs originating from $\mathcal{T}_{h-1}[i]$ and containing keyword k. (b) the values $(\mathtt{S}_k.\mathtt{x}, \mathtt{S}_k.\mathtt{y}, \mathtt{S}_k.\mathtt{z})$ are the sum of the Cartesian coordinates of each microblog containing k and mapped to that cell. These synopses are sufficient to estimate the spatial focus of keywords. Next, we will show how to update their values as new microblogs are inserted at snapshot t and others are eliminated when their snapshots expire.

Spatial Index Update

Updating the tree $\mathcal{T}$ is essential to ensure an up-to-date state of the maintained synopses. The update procedure is triggered once a new snapshot t elapses and involves the *insertion* and the *deletion* of the microblogs published during t and $t-c$, respectively.

Insertion. For each microblog m_i containing at least one keyword, i.e., $m_i \in \mathcal{W}_k^t \wedge k \in K^t$, the tree is traversed from the root to the corresponding leaf node based on the geo-coordinates $m_i.loc_i$. The value $count_T$ of each traversed node is incremented by 1. When the leaf node is reached, $\mathtt{S}_k.\mathtt{n}$ is incremented by 1 and the remaining elements $(\mathtt{S}_k.\mathtt{x}, \mathtt{S}_k.\mathtt{y}, \mathtt{S}_k.\mathtt{z})$ are updated as follows

$$\mathtt{S}_k.\mathtt{x} \mathrel{+}= cos\left(loc_i.lon \times \frac{\pi}{180}\right) \times cos\left(loc_i.lat \times \frac{\pi}{180}\right) \tag{4.15}$$

$$\mathtt{S}_k.\mathtt{y} \mathrel{+}= cos\left(loc_i.lon \times \frac{\pi}{180}\right) \times sin\left(loc_i.lat \times \frac{\pi}{180}\right) \tag{4.16}$$

$$\mathtt{S}_k.\mathtt{z} \mathrel{+}= sin\left(loc_i.lon \times \frac{\pi}{180}\right) \tag{4.17}$$

These equations convert the longitude/latitude coordinates of microblog m_i into the Cartesian coordinate system and accumulate the results into their respective elements $(\mathtt{S}_k.\mathtt{x}, \mathtt{S}_k.\mathtt{y}, \mathtt{S}_k.\mathtt{z})$.

However, microblogs pertaining to a certain event might contain more than one keyword. Hence, the tree will be traversed more than once to insert the keywords of one microblog. To overcome this, we build a temporary inverted index having the microblog ids as keys and the keywords as postings. This helps reduce the time needed for insertions.

Deletion. Similarly, the microblogs $\{\mathcal{W}_k^{t-c}\}_{\forall k \in K^t}$ of the expired snapshot $t-c$, which contain at least one keyword, are used to traverse the tree and to remove the corresponding statistics. This time, however, the statistics of these microblogs are subtracted from the maintained synopses. For the same efficiency reason, a temporary inverted index is created so that the tree is traversed once for each microblog. To improve the efficiency of the deletion process, each time a new node is visited, $count_T$ is decremented by 1. If it becomes 0, the current node and all its children are removed from the tree.

4.6.3 Focus Estimation

In this section, we use the pyramid structure along with the maintained synopses in order to incrementally identify local keywords K_{local}^t and to estimate the focus $\mathcal{F}_k$ of each $k \in K_{local}^t$. Algorithm 4.3 details the steps needed to accomplish these tasks for each $k \in K^t$. These steps can be categorized into: (1) count initialization (Line 3), (2) count update and locality check (Lines 4-21), and (3) focus estimation (Lines 22-24).

Count initialization. Recall that each node in the tree maintains the field $count_K$ to store the number of microblogs containing keyword k. In this step, a value is assigned to $count_K$ of each node in $\mathcal{T}$. For leaf nodes, these counts can simply be initialized as $count_K = \mathtt{S}_k.\mathtt{n}$. Then, the tree $\mathcal{T}$ is traversed recursively to accumulate these counts and propagate the sums up the tree hierarchy until the root node is reached. We end up having the field $count_K$ of each internal node holding the sum of corresponding fields in its children, i.e.,

$$\mathcal{T}_i[j].count_K = \sum_{g \in \mathcal{T}_{i+1}^j} g.count_K \qquad \forall i \in \{0, \ldots, h-2\} \tag{4.18}$$

Count update and locality check. In Line 4, n refers to the index of the cell that is assumed to be a candidate location for the localized event described by keyword

Algorithm 4.3: Locality Check and Focus Estimation

Input: Keywords K^t , tree $\mathcal{T}$, entropy threshold ϕ
Output: Local keywords K^t_{local}, focus $\mathcal{F}_k$ foreach $k \in K^t_{local}$

$K^t_{local} \leftarrow \{\}$
foreach $k \in K^t_{local}$ **do**
  $initializeCounts(k)$
  `// index of root node`
  $n \leftarrow 0$
  $isLocal \leftarrow true$
  for $l = 1$ *to* $h-1$ **do**
    $sum \leftarrow 0$
    foreach $g \in \mathcal{T}_l^n$ **do**
      $\mathcal{S}_k^g := \frac{g.count_K}{g.count_T}$ `// spatial signal estimation`
      $sum \leftarrow sum + \mathcal{S}_k^g$
    $\mathcal{S}_k \leftarrow \mathcal{S}_k / sum$ `// signature normalization`
    $\hat{\mathcal{S}}_k \leftarrow signatureAdjustment(\mathcal{S}_k)$ `// Section 4.5`
    $ent \leftarrow 0$
    foreach $(g \in \mathcal{T}_l^n) \wedge (\hat{\mathcal{S}}_k^g > 0)$ **do**
      $ent \mathrel{+}= \hat{\mathcal{S}}_k^g \times \log_2(\hat{\mathcal{S}}_k^g)$
    `// the entropy of keyword` k `at level` l
    $entropy \leftarrow -1 \times ent$
    if $entropy \leq \phi$ **then**
      $n \leftarrow \arg\max_{g' \in \mathcal{T}_l^n} (\hat{\mathcal{S}}_k^{g'})$
    else
      `// in case keyword` k `is identified as non-local`
      $isLocal \leftarrow false$
      $break$
  if $isLocal == true$ **then**
    `// if keyword` k `is local, it will be added to` K^t_{local}
    $K^t_{local} \leftarrow K^t_{local} \cup k$
    $\mathcal{F}_k \leftarrow geomid(k, \mathcal{T}_{h-1}[n])$

k. The index n is initialized to 0 referring to the root at level 0. Then, the tree is traversed top-down and level-wise starting from the coarsest level (i.e., level 1) to the finest one (level $h-1$) to locate the leaf node (cell) with the highest density in case keyword k is identified as local. In order to consider keyword k a local keyword, it should pass the locality check at each level. Before checking the locality of k at level l using the entropy-based method discussed in Section 4.4.2, the spatial signature of

keyword k is estimated, normalized (Section 4.4), and adjusted (Section 4.5) as can be seen in Lines 7-12. Then, the entropy of the adjusted signature over the entire set of cells $\mathcal{T}_l^n$ is computed (Lines 13-16). If it falls in the range $[0, \phi]$, then keyword k is considered local at level l and the index of the cell having the highest density is returned and stored in n. This will prune the spatial space, because only the tree branch of this cell will be handled in the next iteration (at level $l+1$).

Focus estimation. Once keyword k is recognized as local, it will be added to K_{local}^t (Line 23), and the focus of k (Line 24) is computed as follows

$$\mathcal{F}_k.lon = atan2\left(\frac{\mathtt{S}_k.\mathtt{y}}{\mathtt{S}_k.\mathtt{n}}, \frac{\mathtt{S}_k.\mathtt{x}}{\mathtt{S}_k.\mathtt{n}}\right) \times \frac{180}{\pi} \tag{4.19}$$

$$\mathcal{F}_k.lat = atan2\left(\frac{\mathtt{S}_k.\mathtt{z}}{\mathtt{S}_k.\mathtt{n}}, \sqrt{\left(\frac{\mathtt{S}_k.\mathtt{x}}{\mathtt{S}_k.\mathtt{n}}\right)^2 + \left(\frac{\mathtt{S}_k.\mathtt{y}}{\mathtt{S}_k.\mathtt{n}}\right)^2}\right) \times \frac{180}{\pi} \tag{4.20}$$

where the middle point of the Cartesian coordinates accumulated in $(\mathtt{S}_k.\mathtt{x}, \mathtt{S}_k.\mathtt{y}, \mathtt{S}_k.\mathtt{z})$ is computed and converted back into lon/lat coordinates.

At snapshot t, each local keyword $k \in K_{local}^t$, along with its focus $\mathcal{F}_k$, is used to form an *event entity* (entity for short), denoted by $ee = (k, \mathcal{F}_k, t)$. These entities are appended to a queue called *Event Queue* (*EQueue*) that holds entities generated during $\mathcal{W}$. In the next section, we show that once an entity is appended to this queue, it will be immediately assigned to a group (cluster) that relates to a certain event or labeled as noise.

One last issue we consider is when multiple events on the same topic are taking place at the same time over G. In this case, some shared keywords, referred to as *topical keywords*, are distributed broadly over space, and hence, are mistakenly identified as non-local. For example, when two soccer matches occur simultaneously, topical keywords like "match" and "goal" will be mentioned at the location of both events. To recover the locality of such keywords, we exploit the spatial co-occurrence between topical and local keywords K_{local}^t. The intuition here is that a keyword identified as non-local might pertain to one or more localized events if it shows a certain level of co-location with at least one of the events' local keywords. For this, we employ the Pearson correlation coefficient as detailed in Algorithm 4.4.

In Line 1, the notation T refers to the set of all possible topical keywords. They are traversed to extract those that can be recovered as local keywords (Lines 3-10).

Algorithm 4.4: Topical-to-local keyword recovery procedure.

Input: Keywords K^t, local keywords K^t_{local}, spatial signatures $\hat{\mathcal{S}}_k$ $\forall k \in K^t$, and a threshold γ.
Output: Updated *EQueue*

```
1  T ← K^t − K^t_local // set of topical keywords
2  cells ← {}
   // only considering geo-tagged microblogs
3  foreach tk ∈ T do
4      foreach lk ∈ K^t_local do
5          if cells ∩ cellOf(F_lk) == {} then
6              if ρ_tk,lk ≥ γ then
7                  ee ← (tk, F_lk, t)
8                  EQueue.append(ee)
9                  cells ← cells ∪ cellOf(F_lk)
10     cells ← {}
```

The correlation between the two probability distributing $\hat{\mathcal{S}}_{tk}$ and $\hat{\mathcal{S}}_{lk}$ is measured using the Pearson correlation coefficient defined as

$$\rho_{tk,lk} := \frac{\text{cov}(\hat{\mathcal{S}}_{tk}, \hat{\mathcal{S}}_{lk})}{\sigma_{\hat{\mathcal{S}}_{tk}} \times \sigma_{\hat{\mathcal{S}}_{lk}}}, \tag{4.21}$$

where $\text{cov}(\hat{\mathcal{S}}_{tk}, \hat{\mathcal{S}}_{lk})$ is the covariance that is estimated as $E[(\hat{\mathcal{S}}_{tk} - \mu_{\hat{\mathcal{S}}_{tk}})(\hat{\mathcal{S}}_{tl} - \mu_{\hat{\mathcal{S}}_{tl}})]$ and $\sigma_{\hat{\mathcal{S}}_{tk}}, \sigma_{\hat{\mathcal{S}}_{lk}}$ are the standard deviation of $\hat{\mathcal{S}}_{tk}$ and $\hat{\mathcal{S}}_{lk}$, respectively. The correlation coefficient yields a value in the range $[-1, 1]$, where 1 implies a perfect positive correlation and -1 means a perfect negative correlation. If the correlation between the signatures of the topical keyword tk and the local one lk is above or equal to a certain threshold γ, tk is considered local, and thus, a new event entity ee is generated and appended to *EQueue* (Lines 5-8). The set *cell* holds the geographic cells that the topical keyword has been assigned to so far, which prevents generating multiple entities that have a focus in same cell and significantly improve efficiency as only a subset of the local keywords are compared against the topical keyword tk.

4.7 Event Cluster Generation and Scoring

At snapshot t, a number of event entities that are assumed to refer to some localized events are generated. However, until now it is not obvious which entity belongs to

which localized event. Our focus in this section is to group (cluster) related event entities in order to generate potential localized events and to score them.

4.7.1 Spatial Clustering

According to the first low of geography by Tobler [149], which states that "Everything is related to everything else, but near things are more related than distant things", spatially-close event entities are grouped together (clustered) to form potential localized events. To account for the streaming nature of microbloging services and for the arbitrarily-distributed entities over space, we need the clustering algorithm to fulfill the following requirements:

(1) **density-based**: This enables generating clusters based on the density distribution of entities over space, whereby generated clusters can have arbitrary shapes reflecting the spatial spread of real-world events. Since the number of localized events is unknown, density-based clustering is a good choice as it does not require specifying the number of clusters a priori. Moreover, density-based clustering is more resistant to noise [73], which is an important property of clustering when dealing with messages that are heterogeneously distributed over space.

(2) **incremental**: The streaming nature of published microblogs requires the adoption of a single-pass clustering algorithm, so that once an entity is assigned to a certain cluster, it would not be revisited again.

With these clustering requirements in mind, we chose to apply *IncrementalDBSCAN*[59] to incrementally cluster the entities appended to *EQueue* and form potential localized events $(e_1, e_2, \cdots)$. In Section 4.7.1, we will present the basic concepts and notations important to describe the clustering mechanism underlying DBSCAN. Then, the update operations of the IncrementalDBSCAN algorithm are addressed in Section 4.7.1.

Basics of DBSCAN

To form a cluster using the density-based clustering algorithm, DBSCAN, at least one cluster member should have at least $MinPts$ neighbors within the circular buffer of radius Eps, where $MinPts$ and Eps are the only required parameters.

Given a set of objects D that are to be clustered using DBSCAN. An object p is said to be *directly density-reachable* from object q w.r.t. $MinPts$ and Eps if:

(1) $p \in N_{Eps}(q)$: where $N_{Eps}(q)$ is a subset of D, which contains objects in the Eps-neighborhood of q.

(2) $|N_{Eps}(q)| \geq MinPts$.

On the other hand, an object p is *density-reachable* from q w.r.t. $MinPts$ and Eps if there is a chain of objects $p_1, p_2, \cdots, p_n$ where $p_1 = q, p_n = p$ such that $p_{i+1} \in D$ is directly density reachable from $p_i \in D$. Two "border objects" of a certain cluster might not be density reachable from each other due to the lack of enough objects in their Eps-neighborhoods. To capture this indirect relationship between border objects, the notion of *density-connectivity* was introduced. An object p is density-connected to an object q if there is an object $r \in D$ such that both p and q are density-reachable from r[1]. Therefore, as illustrated in Figure 4.8, object r acts as an intermediate object that connects the objects p and q if both are border objects in the same cluster.

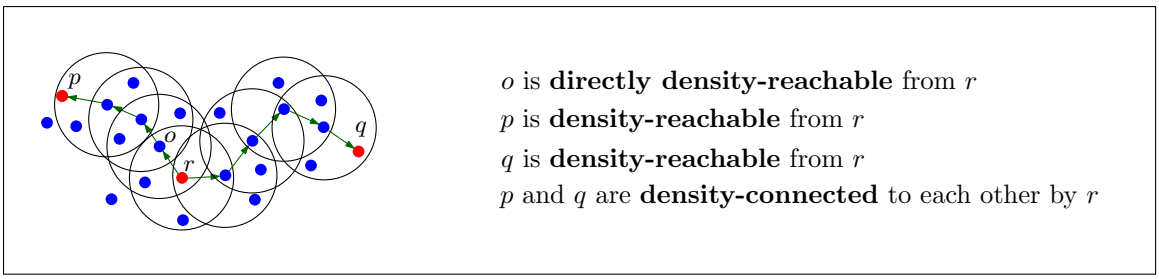


Figure 4.8: An example describing the basic concepts of DBSCAN: density-reachability and density connectivity.

In the context of event detection, a potential event cluster generated by a density-based clustering algorithm consists of a set of density-connected entities[2], which is maximal w.r.t. density reachability. In other words, a cluster C is a non-empty subset of D satisfying the following conditions:

(1) Maximality: $\forall p, q \in D$, if $p \in C$ and q is density-reachable from q, then $q \in C$.

(2) connectivity: $\forall p, q \in C$, p is density-connected to q and vice versa.

There are two different types of entities in a clustering: *core entities* that satisfies the second condition of the directly density-reachable property and *non-core entities* otherwise. The non-core entities are classified into two (1) *border entities* that are not

[1]The phrase "w.r.t. to $MinPts$ and Eps" is omitted from now on whenever it is clear from the context

[2]we replace the notion of objects by event entities (entities for short).

core entities but density-reachable from at least one core entity and (2) *noise entities* that are not core entities nor density-reachable from any other core entities.

First, an entity p is randomly-chosen and all other entities that are in the *Eps*-neighborhood $N_{Eps}(p)$ of p are retrieved. If $|N_{Eps}(p)| < MinPts$, then p is marked as noise. Noisy entities might later be assigned to existing clusters in case they are density-reachable from at least one core object that can be recognized later on. On the other hand, if $|N_{Eps}(p)| \geq MinPts$, then p is a core entity and a new cluster is created. All entities in the *Eps*-neighborhood $N_{Eps}(p)$ of p are assigned to this cluster as well. Then, an iterative process aiming at expanding the cluster with new entities starts. It involves discovering new core objects that are density-reachable from p, identifying non-core objects that are density-reachable from p or other core objects from the same cluster, and assigning all of them (core and non-core entities) to that cluster. This process continues until no further expansion is possible. Then, DBSCAN considers a new unvisited and repeats the previous process again, trying to form a new cluster.

Update Operations in IncrementalDBSCAN

A core characteristic of IncrementalDBSCAN is that it does not revisit the entire set of entities D whenever a new entity is inserted or an expired one is deleted. Ester et al. [59] show that the modifications on some clustering of a database D are limited to a neighborhood of an inserted or deleted entity p. In other words, it is sufficient to apply DBSCAN to the set of entities *affected* by inserting or deleting p. Moreover, it was proved that instead of considering all affected entities when reapplying DBSCAN, only a few number of entities called "seed entities" are considered. Loosely speaking, seed entities are core entities in the *Eps*-neighborhood of other entities that change their core entity property as a result of the update. These seed entities are identified based on the required type of update as follows:

- $UpdSeed_{Ins} :=$
  $\{q | q \text{ is core in } D \cup \{p\}, \exists q' : q' \text{ is core in } D \cup \{p\} \text{ but not in } D \text{ and } q \in N_{Eps}(q')\}$

- $UpdSeed_{Del} :=$
  $\{q | q \text{ is core in } D \setminus \{p\}, \exists q' : q' \text{ is core in } D \setminus \{p\} \text{ but not in } D \text{ and } q \in N_{Eps}(q')\}$

In the following, we detail the insertion and deletion operations of IncrementalDBSCAN. They are applied to the seed entries in order to create new event clusters, to update, or to remove others.

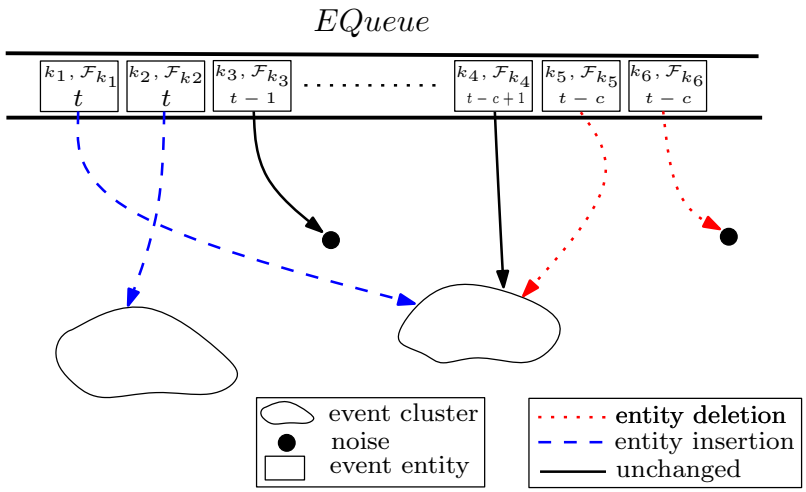


Figure 4.9: *EQueue* holding event entities during $\mathcal{W}$. At snapshot t, new event entities are queued and clustered using IncrementalDBSCAN. Expired entities are removed from *EQueue*.

Entity insertion. Inserting a new entity might lead to establishing new density-connections, while no others are removed. The entities generated at snapshot t are inserted as shown in Figure 4.9. For each inserted entity p, there are 4 possible cases:

(I) Noise: p is considered a noise entity because $UpdSeed_{Ins}$ is empty and there are no *new* core entities after inserting p.

(II) Creation: Before inserting p, the entities in $UpdSeed_{Ins}$ were noise entities. A new cluster containing these noise entity as well as p is created, because these noise entities became core entities after inserting p.

(III) Absorption: The entity p and possibly some noise entities are assigned to an existing cluster cl if $UpdSeed_{Ins}$ is containing core entities that were member of cluster cl before inserting p.

(IV) Merge: The core entities in $UpdSeed_{Ins}$ were members of multiple clusters before the insertion. All these clusters and the entity p are merged into one cluster.

Entity deletion. When deleting an entity p, the density-connectivity property between two or more entities might get lost, but no new density-connections are created. The entities from the expired snapshot $t - c$ are deleted. For each deleted entity p, one of the following cases occurs:

(I) Removal: When $UpdSeed_{Del}$ is empty, p is deleted and other entities in the *Eps*-neighborhood of p change from being assigned to a certain cluster cl to

noise. Also, cluster cl is removed, because cl cannot have core entities outside of $N_{Eps}(p)$.

(II) Reduction: All core entities in $UpdSeed_{Del}$ are density-reachable from each other. In this case, p is deleted and some entities in $N_{Eps}(p)$ might become noise.

(III) Potential Split: deleting a clustered entity p may lead to split the cluster based on the connectivity of p with its neighbors in that cluster.

For more details on entity insertion and deletion in IncrementalDBSCAN, see Ester et al. [59]. Such operations require successive computationally-expensive region queries to retrieve the neighbors of entities. To cope with this, we exploit the spatial index method R*-tree [29].

4.7.2 Cluster Scoring

Given a set of potential event clusters $E^t = \{e_1^t, e_2^t, \ldots, e_{|E^t|}^t\}$ generated at snapshot t or updated from snapshot $t-1$, the goal in this section is to assign a score to each. Assigning a score for cluster e^t is important, because it captures two aspects:

(1) *Cluster Significance.* The higher the score, the more likely cluster e^t refers to an event cluster, i.e., a cluster referring to a real-world event. Building a supervised classifier that can distinguish between event and non-event clusters, e.g., a cluster of keywords mentioned at an airport, is out of the scope of this chapter. However, we will show in the experiments that this score is a useful feature for such a classifier.

(2) *Cluster Evolution.* The score of e^t is computed at each snapshot, which signifies its progress over time.

Before discussing how to score clusters, we present some properties describing the state of a cluster e_i^t:

1. **Start Time** $e_i^t.st$: It represents the snapshot at which the cluster was created.

2. **Keyword Descriptors** $e_i^t.Y$: The entities forming cluster e_i^t are aggregated based on their keywords. This results in a set of tuples referred to as *keywords Descriptors*. Each descriptor $e_i^t.Y[k] = (accWeight, EE)$ belongs to a certain keyword k and consists of the accumulated sum of weights (discussed in Chapter 3) that keyword k got during $\mathcal{W}$, and the keyword's entities assigned to e_i^t

during $\mathcal{W}$, respectively. The entities EE are chronologically ordered by maintaining them in a queue as they arrive, which is important to access the most recent entity in constant time. The entire set of entities in cluster e^t is denoted by $e_i^t.EE$.

3. **Previous Cluster Size** $e_i^t.prevSize$: It refers to the number of entities assigned to cluster e_i^t at snapshot $t-1$. When e_i^t receives more event entities at snapshot t than the entities it loses at the expired snapshot $t-c$, its size increases indicating that the corresponding event is still taking place, yet is getting more popular. Therefore, maintaining the size of e^t at the previous snapshot helps in controlling the evolution of its score based on the change in the cluster size as discussed below.

The cluster scoring scheme we propose relies on the keywords a cluster contains during $\mathcal{W}$. Thus, the score $score(e_i^t)$ of cluster e_i^t at any snapshot depends on the number of keywords it contains and the weights of these keywords. First, we present how to score each keyword in a cluster, and then, how to aggregate the scores of individual keyword to generate a score for the cluster.

To assign score for individual keywords, we assume that the score $score(k, e_i^t)$ of k depends on the following factors:

(1) **Power**: The weight that is computed for keyword k (discussed in Chapter 3) at each snapshot in $\mathcal{W}$ is a useful indication of the significance of k. The more people use this keyword in their microblogs at temporally close snapshots, the higher the weight it obtains. Thus, we define the power of keyword k as the average weight of k in cluster e_i^t during $\mathcal{W}$. More formally,

$$power(k, e_i^t) := \frac{e_i^t.Y[k].accWeight}{|e_i^t.Y[k].EE|}. \tag{4.22}$$

(2) **Presence**: When cluster e_i^t receives keyword k more frequently than other keywords, the confidence that this keyword belongs to the event corresponding to e_i^t should be boosted. Thus, the presence of k in cluster e_i^t is defined as the percentage of times keyword k is clustered to e_i^t during $\mathcal{W}$, i.e.,

$$presence(k, e_i^t) := \frac{|e_i^t.Y[k].EE|}{c}, \tag{4.23}$$

where c is the time window length (measured in snapshots).

(3) **Recency**: It indicates the closeness of the time of the last appearance of k in cluster e_i^t to the current snapshot t. The larger this temporal gap, denoted $tgap(k, e^t)$, is, the lower the score k will obtain. For this, we use an exponential decay function

$$recency(k, e_i^t) := a(1-r)^{tgap(k,e_i^t)} \tag{4.24}$$

where the initial value of a is set to 1, and the decay rate r is set to 0.5. Thus, keyword k will get a maximum recency of 1 if it is clustered in e_i^t at the current snapshot t. Moreover, the recency will decline exponentially by time if k is not being clustered in e_i^t at subsequent snapshots.

Hence, we define the individual keyword score $score(k, e_i^t)$ at cluster e_i^t as the product of the previous factors, i.e.,

$$score(k, e_i^t) := power(k, e_i^t) \times presence(k, e_i^t) \times recency(k, e_i^t), \tag{4.25}$$

which can be reduced to

$$score(k, e_i^t) := \frac{e_i^t.Y[k].accWeight}{c} \times recency(k, e_i^t), \tag{4.26}$$

because the factor $|e_i^t.Y[k].EE|$ in Eq. 4.22 and Eq. 4.23 are canceled out.

Now, the score of cluster e_i^t at snapshot t is defined as the sum of individual scores of the keywords composing e_i^t, i.e.,

$$score(e_i^t) := \sum_{k \in e.Y} score(k, e_i^t) \tag{4.27}$$

By inspection, we noticed that the clusters' scores drop down slowly after their respective events end. In other words, a cluster keeps obtaining high scores for a relatively long period of time even after the event ends. This undesirable phenomenon occurs because: 1) people keep sharing messages with related lkeywords even after an event ends and 2) a relatively large window size is usually chosen to aggregate the sparse spatio-temporal statistics of keywords. To handle this phenomenon, we maintain the size of clusters at the previous snapshot $t-1$ and exploit the Sigmoid function to rapidly lower the their scores as the size decreases over time. Therefore, we redefine the score of cluster e^t as:

$$score(e_i^t) := \frac{\sum_{k \in e_i^t.Y} score(k, e_i^t)}{1 + \exp\{-(|e_i^t.EE| - e_i^t.prevSize)\}}, \tag{4.28}$$

Note that when the number of entities increase over time and the value of $|e_i^t.EE|$ becomes much larger than $e_i^t.prevSize$, then the denominator will approach 1, meaning that the cluster gets the maximum possible score as defined in Eq. 4.27. On the other hand, after the corresponding event ends, the number of entities starts decreasing, leading to a lower score.

Finally, from the definition of a localized event $e_i = (W^t, h^t, start, end)$ discussed in Section 4.3.2, each event component is set as follows. The event keywords W^t are the keyword descriptors of the corresponding cluster e_i^t, i.e., $W^t := e_i^t.Y$. The start time *start* of the event is the creation time of the cluster e_i^t, namely, $start := e_i^t.st$. The end time *end* of the event is set as

$$end := \arg\min_{t'} \{(t' > start) \wedge (score(e_i^{t'}) < score(e_i^{start}))\}, \tag{4.29}$$

meaning that the first snapshot that follows the event's start time *start* and, at which, the cluster gets a score lower than what it has obtained at snapshot *start* is the event end time. The location h^t of e_i is the convex hull of the event entities $e_i^t.EE$, which is computed using Graham Scan algorithm [69].

4.8 Experiments

4.8.1 Experimental Setup

Datasets. We used the three Twitter datasets presented in the previous chapter: **MAD**, **UKR**, and **NYC**, which contain tweets published from the city of Madrid, Ukraine, and highly- populated parts of New York, respectively. For more details on the time intervals and other aspects related to these datasets, see Section 3.7.1. These datasets were created using our TWIPA[1] (TWItter rePository mAnager) tool. TWIPA is a pipeline framework in support of digesting the huge amount of Twitter content and transferring the demanded tweets from files to a fast-access and indexable database, namely a MongoDB-based database.

As can be inferred, the datasets differ in terms of the average number of tweets per hour and the covered geographic areas, which allows us to test LocEvent using datasets having different spatio-temporal sparsity. The tweets are ordered by their timestamps and sent to the framework as a stream of messages, imitating the streaming nature of Twitter.

[1] `http://dbs.ifi.uni-heidelberg.de/index.php?id=twipa`

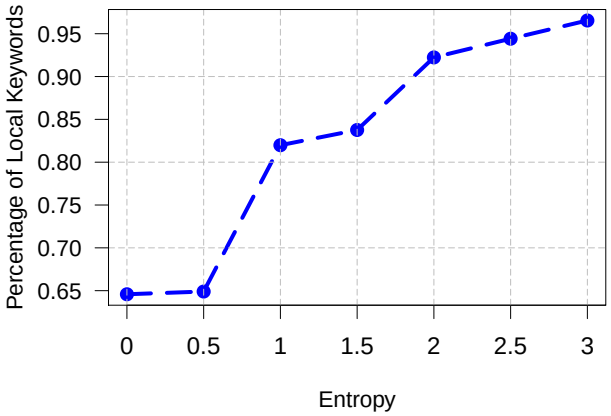


Figure 4.10: The impact of ϕ on identifying local keywords.

Parameter Setting. The length of the used snapshot and sliding window c were set to 1 hour and 6 hours, respectively. We set the cell width of the used grid to 1.1 km. The locality threshold ϕ for this and the following experiments is set as follows. We plot the percentage of keywords recognized as local over a number of locality thresholds. As can be seen in Figure 4.10, a significant change (about 17%) in the percentage of local keywords is observed when varying ϕ from 0.5 to 1.0, and thus, we decided to assign 0.5 to ϕ as a cutoff locality threshold. We empirically set the bandwidth H used for the kernel density estimate (see Section 4.5.2) to $(2 \times I_2)$ where I_2 is the 2×2 identity matrix.

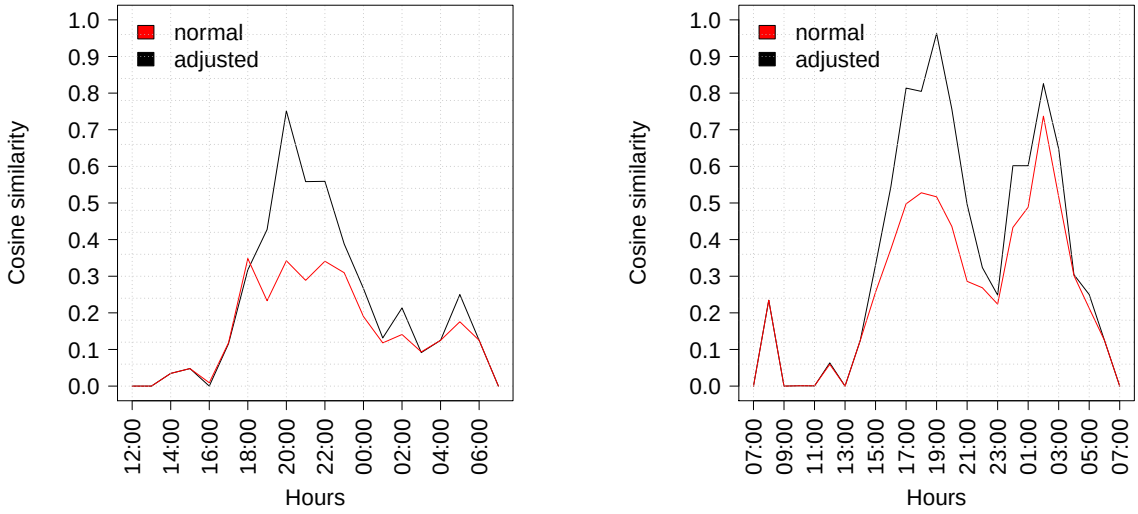
Platform. The experimental evaluation is performed using a platform having an Intel Core i7 (3.40 GHz) with 16 GB main memory. We ran LocEvent under Ubuntu 12.04 (with Java 7 framework).

4.8.2 Evaluation of Spatial Signature

In this section, we test the ability of LocEvent to provide an adjusted spatial signature for each keyword. As for the graph-based regularization technique, we generated a base signature $\mathcal{S}_{base}$ to reflect the typical spread of a localized event. Then, we computed the cosine similarity between that $\mathcal{S}_{base}$ and the signature of relevant keywords. In the context of spatial signatures, this cosine check will result in values in the range [0,1] with the value of 1 being an exact match.

From dataset **MAD**, we considered the event of the "classico" match, and chose a set of relevant keywords (estadio, bernabu, santiago, elclasico, empieze, hastaelfinalvamosreal, madrid, barca). Then, we generated $\mathcal{S}_{base}$ such that $\mathcal{S}_{base}[(-3.68, 40.45)] = 1$, where (long: -3.68, lat: 40.45) refers to the cell containing the Santiago Bernabeu Sta-

dium. Similarly, we constructed another base signature for dataset **UKR** referring to the event of the final UEFA 2012 match at the Olympic stadium in Kiev. The chosen keywords for this event are: final, 2012, stadium, kiev, euro2012, zone, euro, olympic. The cosine similarity is computed between the spatial signature of each keyword and the corresponding base signature $\mathcal{S}_{base}$. This similarity check was performed before and after the graph-based spatial adjustment. Then, we took the mean of the resulting similarity values for all specified keywords. This process is conducted during a number of snapshots as shown in Figure 4.11. In this Figure, the adjusted spatial signatures are more similar to the expected $\mathcal{S}_{base}$ than non-adjusted signatures, especially during the events' time. This means that during localized events, adjusted spatial signatures are of more focus on the location of an event, which mitigates the spatial noise and sparsity problems discussed in Section 4.4.



(a) (b)

Figure 4.11: Comparing the spatial signatures of keywords before and after the graph-based signature adjustment. (a) Dataset **MAD** (Madrid), from 2013/1/30 12:00 to 2013/1/31 7:00. (b) Dataset **UKR** (Ukraine), from 2012/7/1 7:00 to 2012/7/2 7:00

The spatial signature of a certain keyword can be visualized on a map before and after the graph-based adjustment as shown in Figure 4.12. It depicts the spatial distribution of keyword "kiev" aggregated over one-hour snapshots from (17:00 2012/7/1) to (2:00 2012/7/2). This word refers to the city where the final EUFA 2012 match took place. The spatial distribution is expected to be focused on (and close to) the stadium of the match. However, and as can be seen, the spatial signals of this word are distributed over many cells around the stadium, which can be justified by the effect spatial outliers. After performing signature adjustment for this keyword,

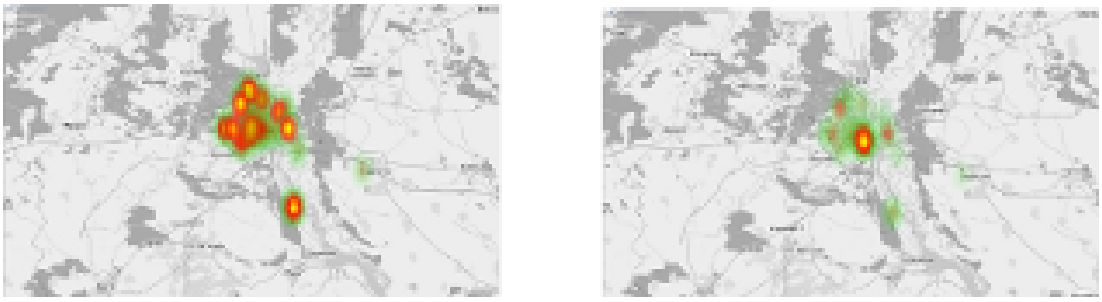


Figure 4.12: Visualizing the spatial signature of word "kiev" before the graph-based signature adjustment (left) and after adjustment (right).

an updated signature is obtained with more focus on the match stadium as shown in the right part of the figure.

To test the impact of the gazetteer-based signature regularization in providing a reliable decision about the locality of keywords, we chose some events from Table 3.3 and collected statistics pertaining to the considered keywords before and after eliminating spatial outliers. In Table 4.1, the second and third column show how the entropy mean of the keywords decreases drastically after eliminating spatial outliers, which, of course, increases the chance of identifying them as local keywords. The forth column illustrates the percentage of *spatially-adjusted keywords* that become local and have a focus at the event location after the adjustment. The more important and popular the localized event, the higher is the percentage. The fifth column shows the precision of the adjusted keywords, i.e., these keywords that actually refer to the corresponding event. As a result, the majority of the adjusted keywords that have a focus at the event location pertains to that event.

Table 4.1: The performance of spatial outliers elimination using the gazetteer-based regularization.

EID	**Before Adjust.**	**After Adjust.**	**Spatially-adjusted**	**Precision**
$E6$	0.803	0.1603	56.2%	$\frac{70}{104} = 0.67$
$E8$	0.49	0.028	39.6%	$\frac{16}{20} = 0.8$
$E9$	0.478	0.027	41.4%	$\frac{41}{56} = 0.732$
$E10$	1.345	0.807	33.6%	$\frac{34}{40} = 0.85$

In Figure 4.13, the keyword "krewella" of event $E8$ is considered and the temporal profile of the keyword's entropies during 10 1-hour snapshots is plotted. We notice that the entropy values increase over time due to the fact that more and more people will be aware of the event occurrence by time through TV, friends etc. After elimination of outliers (discussed in Section 4.5.1), these values become smaller (below ϕ

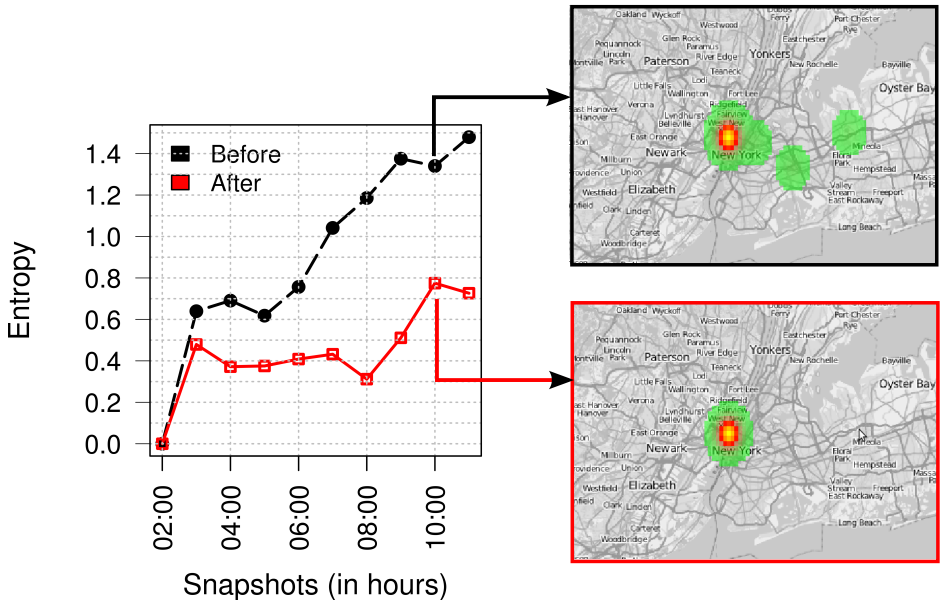
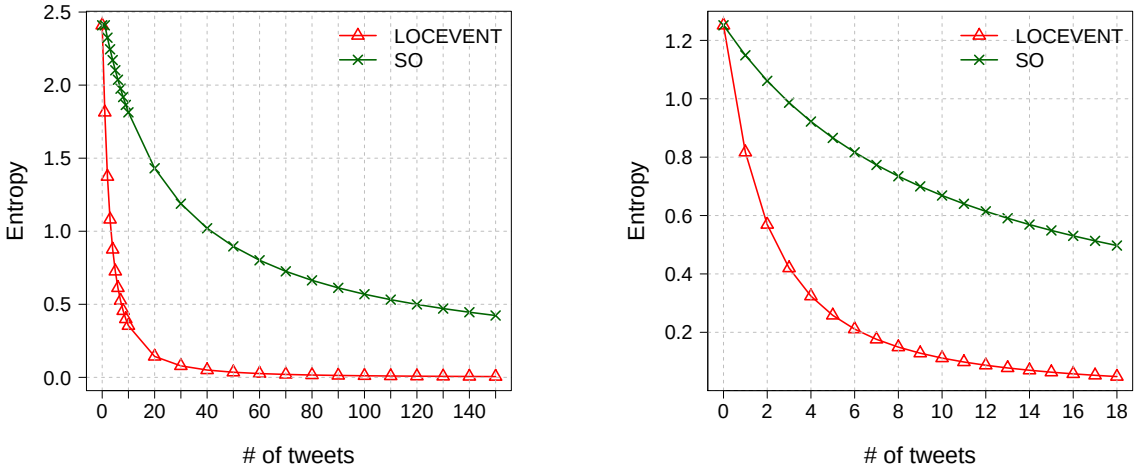


Figure 4.13: The impact of outliers elimination using gazetteer-based regularization for keyword "krewella" of event $E8$ during 10 1-hour snapshots.

most of the time), and thus, the impact of these outliers on the spatial distribution of keyword is minimized.

4.8.3 Keyword Locality Evaluation

We evaluate another important aspect of LocEvent, which is how much information is needed to handle spatial outliers and regain the expected spatial distribution of keywords. For this, a number of tweets are injected into dataset **NYC** at snapshot (2013/11/13 03:00), having the geo-coordinates (lon: -73.983, lat: 40.738) of an assumed event location. These tweets contain the words: "fire", "shooting", and the georeference "pizza pub" that is close to the event location. In Figure 4.14, we plot two diagrams for both keywords, and show how the entropy decreases as the number of such tweets increases. To highlight the value added by LocEvent, we compare its performance against a baseline that does not perform outlier elimination, denoted as `SO`. In Figure 4.14a, it is observed that 120 tweets are required to make the keyword "fire" local using `SO`. This is because this word is usually mentioned in different contexts, and thus, it has a broad spatial coverage. However, LocEvent entails only 8 tweets to reach the locality threshold ϕ. On the other hand, and as can be seen in Figure 4.14b, the keyword "shooting" has reached ϕ only with 3 tweets using LocEvent since it is sparsely used over space.

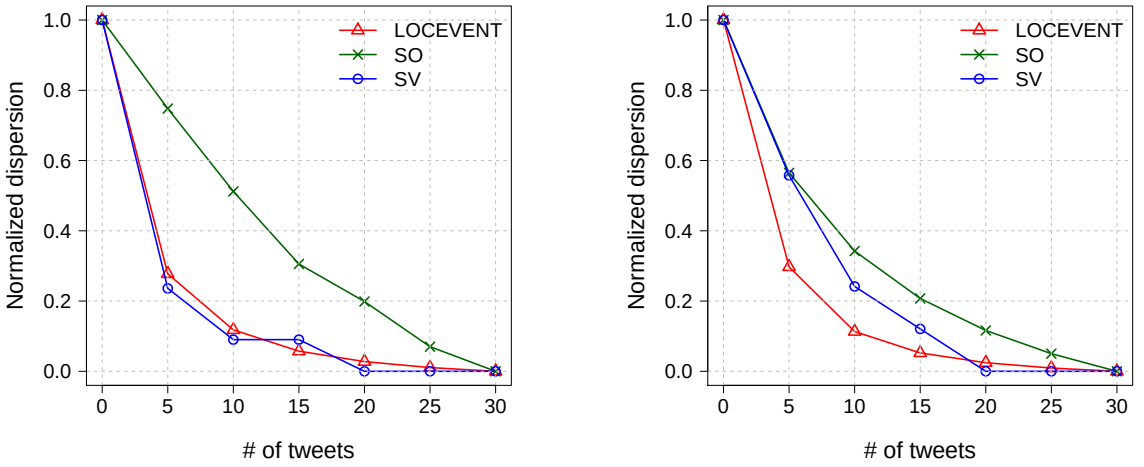


(a) fire (b) shooting

Figure 4.14: Impact of increasing the number of a keyword's occurrences on entropy using LocEvent and SO.

In addition, we compare LocEvent against the model of Backstrom et al. [21], which we denote as SV. SV is a probabilistic model that captures the spatial variation of geo-tagged search queries and gives an estimation of the central location of a query and its spatial dispersion. The dispersion indicates whether the query has a local interest or broader regional or national appeal, and is quantified by estimating a value for the dispersion parameter α. The larger α, the faster the query decays away from the center and the more localized it is. In this context, we consider the preprocessed content of a tweet as a query in order to be able to utilize SV and compute the dispersion of a particular keyword. Figure 4.15 shows the effects of varying the number of injected tweets from 0 to 30 on a normalized versions of $-\alpha$, the entropy from LocEvent, and the entropy from SO. In Figure 4.15a, both LocEvent and SV exhibit a similar and fast response to the increase in the number of tweets containing the keyword "fire". Both approaches outperform SO, because SO relies only on a word's spatial distribution whose entropy decreases slowly in the case of widely-used keywords such as "fire". Figure 4.15 shows that SO has a better response to the increase in tweets containing the keyword "shooting" due to its sparse spatio-temporal usage. However, LocEvent outperforms both SO and SV, which highlights its effectiveness in providing a rapid locality detection for a sparsely-used keyword.

Finally, we compare the accuracy of LocEvent in computing the spatial focus of local keywords against SO and SV. For this, we chose some events from Table 3.3, computed the focus of each keyword using the three approaches, and plotted the average distance error (see Figure 4.16). A keyword k focus obtained from the baseline



(a) fire (b) shooting

Figure 4.15: Effect of the number of tweets on normalized dispersion measures from LocEvent, SO, and SV.

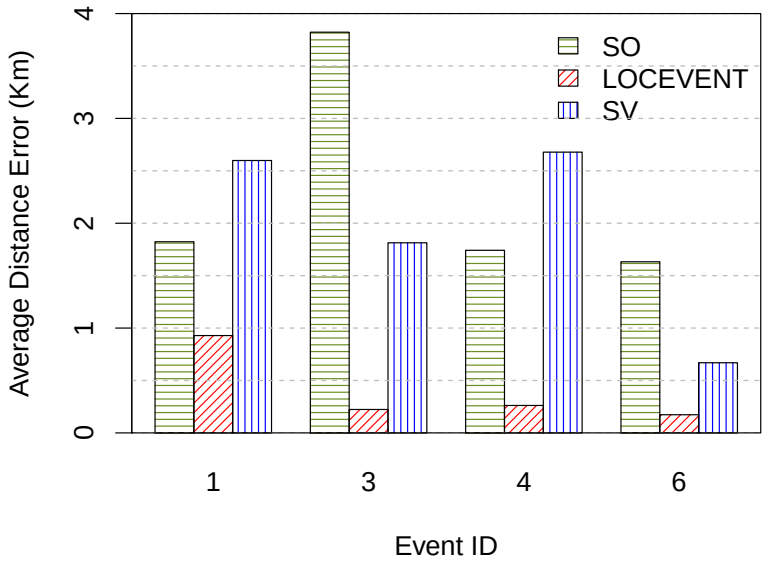


Figure 4.16: Comparison between LocEvent, SO, and SV in terms of the average distance error for a number of events.

approach SO is estimated from the tweets containing k by computing the weighted center of gravity of their geo-coordinates. To calculate the error distance for a particular keyword, we manually specify a location for each localized event as the central geographic point of the event venue, and then apply the Haversine formula [141] to determine the distance between the keyword focus and the event's central point. Figure 4.16 shows that LocEvent outperforms the other approaches in estimating an accurate focus for all events, which can be explained by the fact that LocEvent not only eliminates outliers, but also preserves a location summary for each keyword

inside each cell. The approaches `SO` and `SV` exhibit an almost similar behavior, and, of course, spatial outliers lead to relatively large error distances.

4.8.4 Social Evidence Evaluation

In this section, we present our evaluation of the method discussed in Section 4.5.3 to identify social evidences. Table 4.2 shows the top 5 scored tweets at snapshot 2012/7/1 20:00 from dataset **UKR**. Each row corresponds to a single tweet showing its content, geo-location and score. The geo-location represents the lower left corner of the cell from which a tweet originates. The listed tweets touch the topic of a soccer match. These tweets got the highest scores since (1) they contain keywords (e.g., UEFA, EURO, Final etc.) that are relevant to the final UEFA 2012 match, and (2) they originate from the central cells (in this case, the cell of the stadium and its neighbors) of the match.

Table 4.2: Top-5 scored tweets extracted from dataset **UKR** during snapshot 2012/7/1 20:00

Tweet	**(long, lat)**	**Score**
España <3 @ Official Fan Zone of UEFA EURO 2012 http://t.co/XYiMePtF	30.52, 50.44	0.86
Go Spain @ НСК Олімпійський / NSC Olimpiyskiy http://t.co/8YBUkOgD	30.52, 50.43	0.84
Euro 2012 final. @ НСК Олімпійський / NSC Olimpiyskiy http://t.co/Kg0ny6FL	30.52, 50.43	0.80
Final euro :D (@ НСК Олімпійський / NSC Olimpiyskiy w/ @2easy4dony) http://t.co/z8quWYim	30.52, 50.43	0.80
I'm at Official Fan Zone of UEFA EURO 2012 w/ @mashalet http://t.co/QW0alqst	30.52, 50.44	0.76

In addition, we computed the score of each tweet posted during the snapshots 15:00, 16:00, ..., 23:00 on July, 1st 2012 from dataset **UKR**. Then, we retrieved the tweets from each snapshot whose scores are greater than or equal to 0.5, and called them *potential social evidences*. These potential evidences were then given to a human annotator who labeled each of them as "evidence" or "non-evidence". We assume that during this time interval the only event occurred was the final EUFA 2012 match, and thus, a tweet is annotated as "evidence" if it talks about this event. After that, we computed the precision at each snapshot as the ratio between the number of tweets labeled as "evidence" and the number of potential social evidences. Figure 4.17

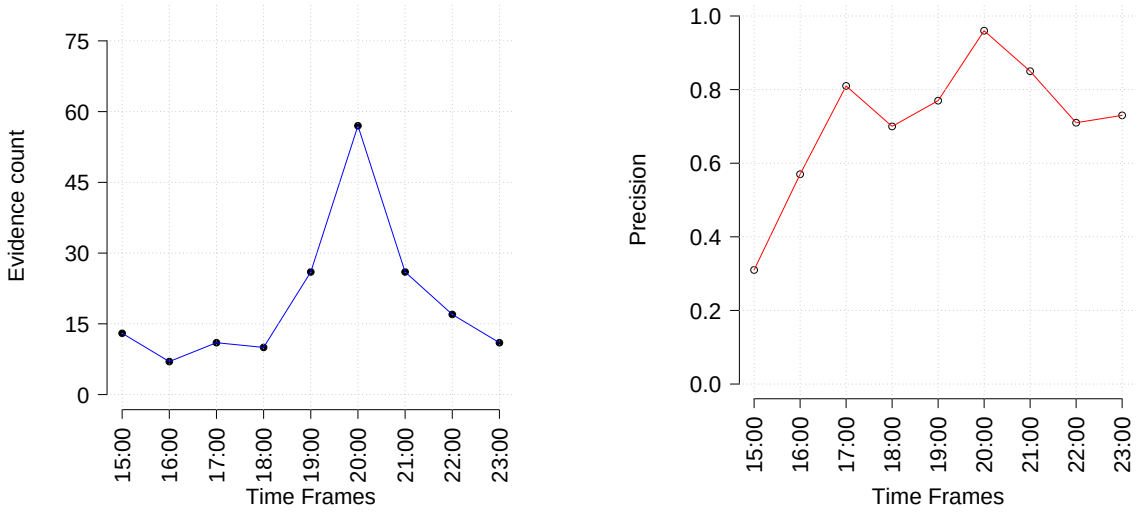


Figure 4.17: (a) The number of potential social evidences per one-hour snapshots. (b) The precision of correctly identified social evidences per each snapshot.

shows that the number of potential social evidences increases when getting close to the start time of the final match. That is, the final match stimulates more users to publish tweets containing related keywords at the cell of the stadium. Furthermore, the precision of tweets that are correctly labeled as "evidence" was at maximum (96%) at snapshot 20:00 (the match started at 20:45 EET).

4.8.5 Overall Performance of LocEvent

Evaluation Metric

The typical performance measures used in this context are *precision* and *recall*. The precision refers to the fraction of actual event clusters to the entire set of generated clusters. Following [103], if two clusters pertain to the same event, both are considered correct in terms of precision. The recall corresponds to the fraction of actual event clusters to the entire set of events that can be extracted from a dataset. While it is easy to estimate the precision, it is not for the recall, because of the absence of a ground truth. In fact, it is not feasible to enumerate all possible events in a dataset trying to build such a ground truth, and thus, we will only focus on the precision. Since the recall is an important metric reflecting the performance of LocEvent in finding a large number of event clusters, we estimated a variation of the recall based on a simplified assumption discussed below. Then, this estimation is exploited to set

the value of one of the clustering parameter $minPts$ that, if carefully chosen, results in a higher precision and a larger number of formed event clusters.

Clustering Parameters

Generating event clusters from entities using IncrementalDBSCAN requires setting two parameters: Eps and $MinPts$. We set Eps to 100m, a relatively small value, in order to (1) ensure the detection of small-scaled events and to (2) reduce the chance of generating non-event clusters from noisy event entities that are sparsely distributed over space. The parameter $MinPts$ has a large impact on precision. Small values of $MinPts$ result in producing a large number of clusters and increase the chance of forming non-event clusters (low precision), and vice versa. Thus, we opt to use the harmonic mean (F-score) that provides a balanced value between precision and recall. Here, we describe how to assign a value to $MinPts$ using F-score in the absence of a ground truth. First, we set $MinPts$ to 10 and ran LocEvent over only a one-day interval from dataset **NYC**, because the manual annotation for the entire dataset on a variety of parameter values is a time consuming process. The chosen date was Nov. 23rd, 2013 because, on that day, the largest number of generated clusters was observed. Then, we ran LocEvent again using other values of $MinPts$, i.e., $MinPts = 15, 20, 25$. As shown in Figure 4.18, increasing $MinPts$ leads to a better precision. To estimate the F-score for each value of $MinPts$, we should have the corresponding recall values, which is theoretically impossible without a ground truth. To cope with this, we took the first configuration $MinPts = 10$ as a baseline and assumed that all event clusters on that day are generated, and hence, we had a recall of 1. Then, we computed the recall using the other values, i.e., $MinPts = 15, 20, 25$ w.r.t. this assumption. The values $MinPts = 10, 15$ yielded the highest F-score. Furthermore, we chose an intermediate value of $MinPts = 12$ that even resulted in a higher F-score; and thus, $MinPts$ is set to 12.

Detection Results

We compare LocEvent against the state-of-the-art local event detection approach *Jasmine* [154] after running both systems over tweets from dataset **NYC**. *Jasmine* represents local events as clusters of georeferences associated with key terms that co-occur with these georeferences. To increase the chance of finding small-scaled events, *Jasmine* uses a gazetteer that provides a mapping between place names and physical locations. To build this gazetteer, place names from geo-tagged tweets are extracted

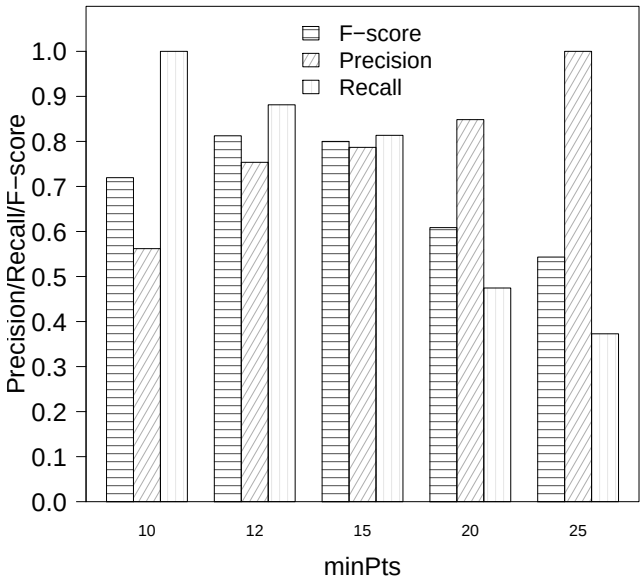


Figure 4.18: The impact of *MinPts* parameter on precision, recall, and F-score.

using a rule-bases NER technique, and then, these names are associated with the mean of the geo-coordinates of tweets containing these place names. To resolve ambiguity, the place names having high geographical variance are ignored as they might refer to ambiguous place names such as "Burger King". We implemented *Jasmine* and ran it over the tweets published during Nov, 23rd. It produced 93 clusters with a precision of 33.33% as shown in Figure 4.19. LocEvent yielded a precision of 75.36% and extracted 8 more localized events than *Jasmine*. This high precision achieved by LocEvent is due to (1) the strict filtering of words to identify the keywords that are both event-related and local and (2) the usage of a density-based clustering algorithm that is noise-tolerant. Moreover, LocEvent produced more event clusters because, unlike *Jasmine*, it does not rely on building clusters around georeferences. However, LocEvent can form clusters anywhere within the geographic space G. After that, we ran LocEvent over the entire dataset **NYC** and extracted 442 distinct events with an overall precision of 68.22%. Interestingly, this precision is promising as LocEvent does not rely on a supervised learning approach to filter out non-event clusters [35], nor on an external data source, such as Wikipedia, as done in [103]. Moreover, we investigated the role that the score of a cluster plays in distinguishing event from non-event clusters. For this, we divided the range of yielded scores into sub-intervals and plotted the aggregated precision. As can be seen in Figure 4.20, the higher the score of a cluster is, the more likely the cluster is referring to a real-world event. For example, all clusters obtained scores larger than or equal to 15 were event clusters.

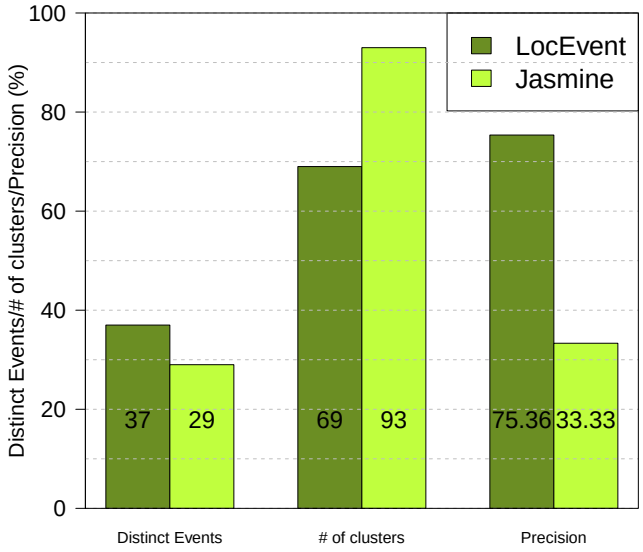


Figure 4.19: Comparison between LocEvent and *Jasmine*.

Therefore, these scores can be used as a useful feature for a supervised classifier to automatically filter out non-event clusters.

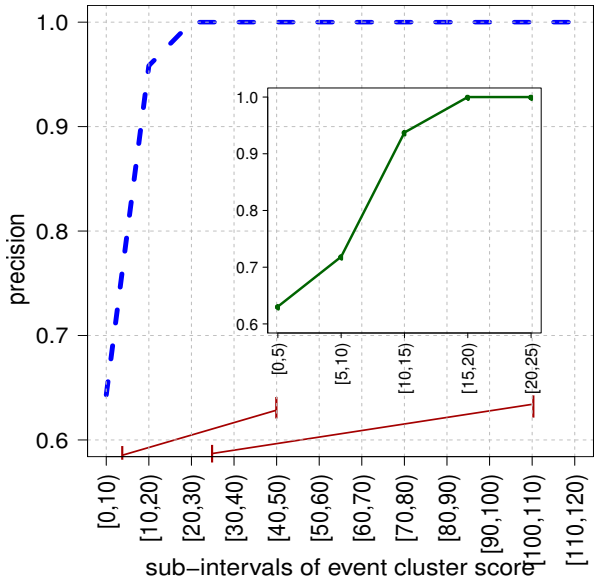


Figure 4.20: The relationship between the scores assigned to clusters and the resulting precision. Higher scores lead to improve precision.

Event Cluster Evolution

LocEvent has an additional advantage over *Jasmine* as it can track the evolution of a detected event by assigning a dynamic score to its respective cluster. Table 4.3 lists

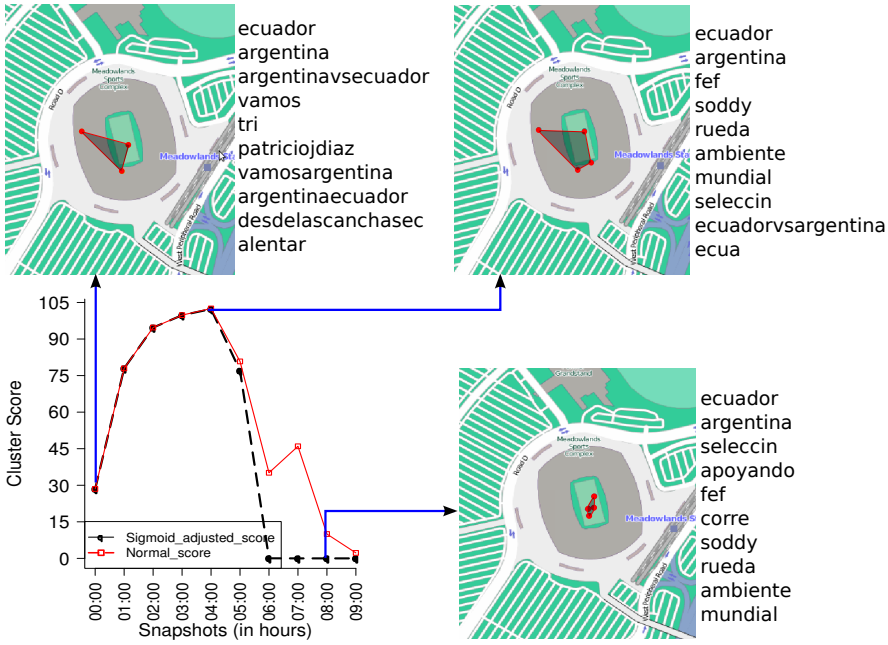
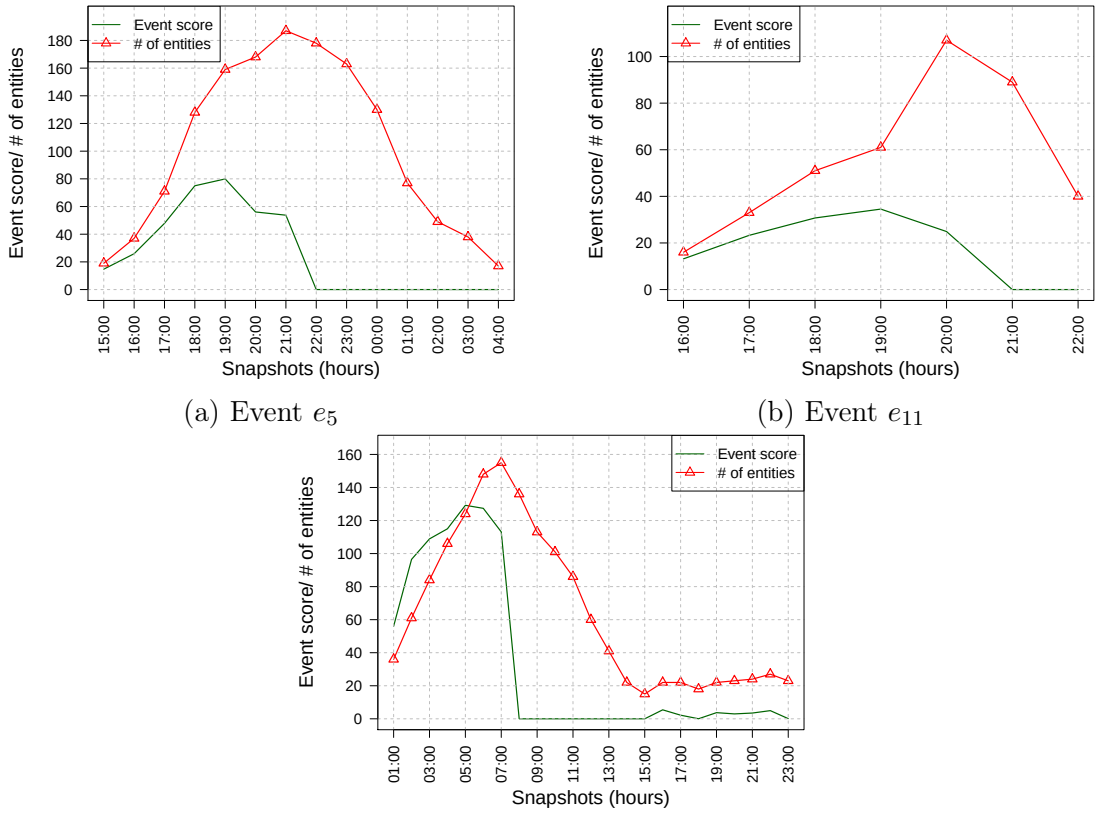


Figure 4.21: The evolution of event e_4 (soccer match between Argentina and Ecuador) over time. The figure depicts the dynamics of the event location, estimated as a convex hull covering a part of the stadium.

the top event clusters with respect to their scores. For each event, we show the top-10 keywords, the highest score the event obtained, its estimated start time (GMT), and a description for the event. Interestingly, these significant events took place at either *Madison Square Garden* or *MetLife Stadium*, where prominent concerts and sport events are hosted, respectively. Moreover, LocEvent, in addition to its ability to extract relevant keywords for each event, can estimate fine-grained location, start time, and end time information. For example, in Table 4.3, e_4 corresponds to the soccer match between Argentina and Ecuador, which took place at *MetLife Stadium* at 11/16 00:30 GMT. The snapshot of the estimated start time is (11/16 00:00), meaning that LocEvent could precisely determine the hour at which the match started. Furthermore, LocEvent captures the dynamics of each cluster by tracking the semantics (also called concept drift) and the spatial evolution of clusters over time. As shown in Figure 4.21, the score of cluster e_4 is updated over time to reflect its changing significance until it diminished. Two versions of the scoring scheme are plotted: the normal and sigmoid-adjusted schemes, discussed in Section 4.7.2. As can be seen, the sigmoid-adjusted version tries to drop the cluster score down quickly after the end time of the soccer match, ensuring a better estimation of the event's time interval.

Table 4.3: A sample from the most significant event clusters identified by LocEvent from dataset **NYC**.

ID	Keywords	Score	Start Time	Description
e_1	yeezus, tribe, quest, kanye, kanyewest, yeezy, garden, madison, kanyemsg, yeezustour	197.363	11/24 23:00	A concert tour by the rapper Kanye West at Madison Square Garden
e_2	timberlake, jtimberlake, justin, justintimberlake, worldtour, shadows, tijucaemny, experiencetour, jttour, fallon	127.373	11/07 01:00	A concert for Justin Timberlake at Barclays Center
e_3	nyr, nyrangers, rangers, thegarden, hat, hattrick, garden, msg, posted, lgr	118.616	11/30 17:00	A hockey game btn Canucks & Rangers @ Madison Square Garden
e_4	ecuador, argentina, fef, soddy, rueda, ambiente, mundial, seleccin, ecuadorvsargentina, ecua	103.405	11/16 00:00	A soccer game btn Argentina and Ecuador @ MetLife Stadium
e_5	saints, orleans, jets, rex, seats, game, jetup, brees, tailgating, nfl	79.935	11/03 15:00	New York Jets vs. New Orleans Saints @ MetLife Stadium
e_6	raiders, giants, nygiants, nyg, oak, rutherford, bigblue, gmen, east, stadium	76.306	11/10 16:00	American football btn Raiders & Giants @ MetLife Stadium
e_7	paramore, macklemore, yelyahwilliams, macklemoreconcert, paramoremsg, crushcrushcrush, macklamore, confetti, tourlife, metric	70.193	11/14 03:00	American rock trio from Franklin, Tennessee.
e_8	dallascowboys, cold, giants, degrees, cowboysnation, tied, cowboys, gmen, freezing, nyg	60.305	11/24 19:00	Dallas Cowboys vs. New York Giants MetLife Stadium
e_9	macklemore, rockets, lewis, linsanity, houston, theater, lin, ryan, yorker, alisonmoyet	50.485	11/14 15:00	an American hip hop duo
e_{10}	staten, island, marathon, ingnycmarathon, runners, whitehall, line, start, ferry, statenislandferry, gooooo	49.648	11/07 01:00	the start line of NYC Marathon @ Staten Island
e_{11}	spurs, nyknicks, knicks, garden, thegarden, manuginobili, knicksspurs, blowout, msg, usaf	34.53	11/10 16:00	Spurs vs. Knicks @ Madison Square Garden



(a) Event e_5

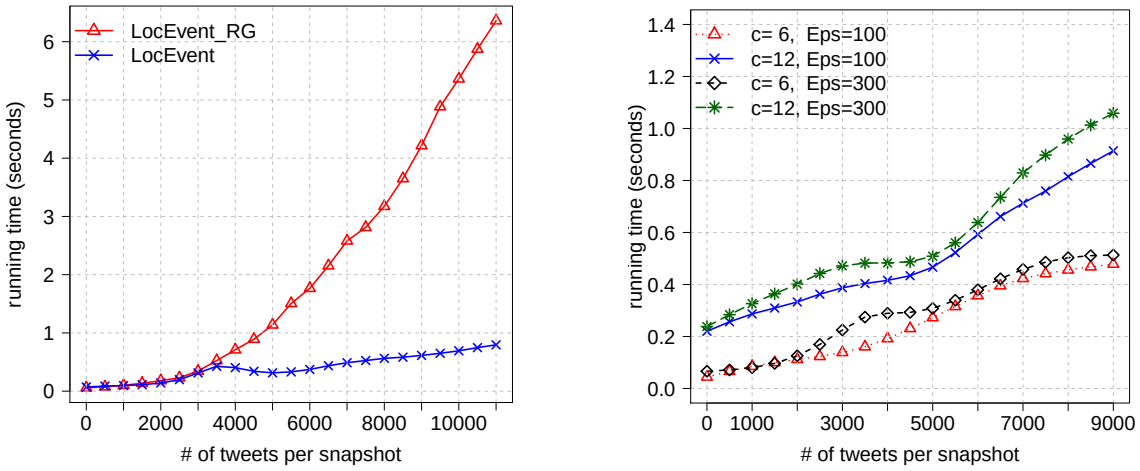
(b) Event e_{11}

(c) Event e_2

Figure 4.22: The temporal evolution of three events described in Table 4.3. Due to the impact of the sigmoid-based score adjustment, a drop in the number of entities leads to a drastic decrease in the score of an event.

In Figure 4.22, the temporal distributions of the scores and the number of event entities pertaining to three events from Table 4.3 are plotted. As can be seen, LocEvent, by tracking the evolution of event clusters, is able to approximate the time interval of an event. For instance, the estimated start time of event e_2 is (11/07 01:00) which is the actual start time of this event, i.e., (11/06 20:00 EST)[1]. The estimated end time of this event is (11/07 08:00). We empirically found that estimating the end time of events is non-trivial as people usually tend to stay at the location of an event for a relatively long time after it has ended and keep publishing about the event. In spite of using the sigmoid-based adjustment method described in Section 4.7.2, there is a need to extract temporal expressions from event-related tweets and use them for a better end time estimation, an aspect we leave for future work, as discussed in Chapter 6.

[1] `http://www.songkick.com/concerts/17479019-justin-timberlake-at-barclays-center`, accessed June 2014



(a) (b)

Figure 4.23: (a) Running time against the number of tweets per snapshot. (b) the impact of window size c and clustering parameter *Eps* on runtime.

Scalability of LocEvent

After running LocEvent over dataset **NYC**, 4.37 minutes are needed to process 3.8 million tweets and generate the scored clusters. This time includes tweets preprocessing, keyword extraction, the four stages of LocEvent, and the index update. Hence, in average, 0.069 ms is required to process each tweet. In Figure 4.23a, we compare LocEvent against LocEvent_rg, a baseline version without spatial indexing. LocEvent outperforms LocEvent_rg, because of its ability to accomplish fast pruning of the space until the event cells event are reached. Figure 4.23b illustrates the impact of the window size (c) and the clustering parameter *Eps* on the computational efficiency of LocEvent. Increasing both parameters results in larger running time. For *Eps*, the increased running time is due to traversing more event entities for a neighborhood check in spite of using the R*-tree index. On the other hand, the window size c shows larger impact on the system's efficiency because (1) more snapshots are considered to compute the current recurrent rate of each word and (2) the spatial index $\mathcal{T}$ will have a larger size as more information is aggregated when using a larger time window.

Another aspect that supports the scalability of LocEvent is that it can be easily parallelized. Fortunately, each word is processed separately in the keyword extraction (Section 3), focus estimation of local keywords stages. This makes it easy to split the set of words $\mathcal{W}^t$ published during snapshot t among multiple processing threads. Each thread processes some words through the first two stages and generates a set of event

entities. Then, the generated entities from all threads are appended to a synchronized version of the event queue *EQueue*.

4.9 Summary and Discussion

In this chapter, we proposed LocEvent, a novel framework to detect localized events from social media streams and to track the spatio-temporal evolution of detected events in an online fashion. LocEvent adopts the sliding window model to conduct a near real-time event detection and tracking. Each time the window slides and a new snapshot elapses, LocEvent follows a 4-stage procedure: (1) Keyword extraction: event-related keywords that exhibit significant increase in usage are extracted using KeyPicker discussed in Chapter 3. (2) Focus of local keywords: local keywords having spatially limited spread are identified and the spatial center (focus) of each local keyword is estimated. (3) Event cluster generation: local keywords are spatially clustered, which generates potential event clusters. (4) Cluster scoring: the generated clusters are finally scored based on the temporal characteristics of the keywords they contain. This type of scoring helps determine the significance of a cluster and how it evolves over time.

LocEvent combines a number of key features, setting it apart form existing approaches.

(1) LocEvent conducts a multi-stage noise elimination procedure. That is, non-event words are screened out in Stage-1 while non-local keywords are eliminated in Stage-2. In Stage-3, we utilize a noise-resilient clustering algorithm and, in Stage-4, clusters are scored so that non-event clusters obtain low scores.

(2) In addition to handling temporal problems, as discussed in Chapter 3, LocEvent tackles two spatial problems, namely, spatial outliers and spatial sparsity, which adversely affect a reliable estimation of spatial signatures. To cope with spatial outliers, we proposed two novel regularization procedures: graph- and gazetteer-based regularization. Spatial sparsity is handled by exploiting a non-parametric kernel density estimate.

(3) To ensure scalability, LocEvent employs a space-partitioning index that is incrementally updated as time progresses by accumulating summarized statistics (synopses). Such synopses are important in identifying local keywords and in estimating their spatial focus. Using this type of indexing, the geographic space can be quickly pruned, resulting in an efficient local keyword identification.

(4) For each detected event, a set of descriptive keywords, a location, and a time interval are estimated.

We have conducted extensive evaluations using datasets collected from Twitter. The experimental results reveal that LocEvent has high precision, when compared to related approaches. Moreover, the results show its scalability, ability to capture the dynamics of events, and robustness against noise and outliers.

Chapter 5

Context-Aware Event Recommendation in Social Media

In the previous chapter, we have discussed the online detection of localized events from a stream of microblogs. For each detected event, our proposed framework extracts a list of descriptive keywords, estimates their locations and time intervals at a fine-grained spatio-temporal resolution. In this chapter, as an application, we consider the personalized event recommendation problem. The aim is to provide users with a ranked list of future events that suit their preferences by proposing a context-aware event recommendation model.

5.1 Introduction

Today, the proliferation of information about social events, disseminated through a large number of location-based social networks (LBSNs) such as Foursquare is driving the development of new services and applications towards a better user interaction. Such event information is either hidden within an immense amount of noisy content and needs to be extracted using some event detection methodology [3, 104, 156], or it is explicitly provided by one of the event-based social networks (EBSNs) such as Meetup[1] and Plancast[2]. Meetup, for instance, provides an online platform for users to create and organize social events that are to be held at some locations [107]. In addition, Meetup enables its members to join virtual groups, where events are created by and announced to their members. Users can then show their intent to attend these events by choosing "yes", "no", or "maybe". On the other hand, the introduction of

[1] http://www.meetup.com/
[2] http://plancast.com/

location-acquisition technologies that are embedded in today's smartphones allow for enriching both events and users with location information. As a consequence, the activity history of users can be analyzed to mine useful user-event attendance patterns, fostering the development of *context-aware* recommendation systems.

Our goal in this chapter is to uncover latent patterns of user interest in attending social events, which are buried in the content of LBSNs and EBSNs. These patterns, in turn, provide personalized recommendations that suit users' tastes. Building recommender systems is a well-established research area that includes a large number of recommendation models adapted to different application domains. Generally speaking, these recommender systems rely on one of the following strategies:

(1) *Content filtering*: It is based on generating a profile for each user and event. For instance, the profile of an event could include characteristics showing its genre, the age range of participants, location, and so forth. Likewise, a user profile might include demographic information, age, gender etc. These profiles are then used to rank the events with respect to a certain user and to recommend, for example, the top-k ranked events to that user. Applying such content-based approaches requires collecting external information that might be unavailable or not easy to collect [92].

(2) *Collaborative filtering*: This strategy only depends on the past behavior of users, which is mined from their attendance to past events. Collaborative filtering [67] uncovers and quantifies the relationships between both users and events towards identifying new user-event associations. One important aspect of collaborative filtering is its domain-free nature in the sense that it provides a generic mechanism to address (latent) data characteristics that are difficult to profile using content-based techniques.

Therefore, collaborative filtering (CF, for short) in general yields more reliable results than content-based techniques [92]. However, it suffers from the *cold-start* problem, meaning that it cannot recommend events to new users who have no (or sparse) event attendance history and fails to suggest future events to them. Additionally, the direct application of CF in the domain of event recommendation has a number of other challenges due to the unique nature of this domain. These challenges are summarized as follows:

(1) Sparse event attendance history. A user can only attend a limited number of social events, which degrades the performance of CF in extracting common patterns for users who have similar event preferences [102].

(2) Events, as items to be recommended, are time-specific. That is, only future events can be recommended to users. Future events usually receive their ratings (or assigned as attended) shortly before they start or after they ended.

(3) Spatio-temporal proximity matters. The temporal distance between the current time and the start time of an event on one hand and the spatial proximity between the location of the user and the event on the other hand jointly contribute to directing the recommendation process.

To this end, we propose a context-aware event recommender system on the basis of model-based CF techniques that are also referred to as *latent factor models*. More precisely, our model is built upon the well-known *matrix factorization* that is one of the most successful realizations of model-based CF. Using matrix factorization, each user and event is characterized by a factor vector whose factors are iteratively estimated by minimizing a cost function. We adapt this model by accounting for the (1) social interactions between users, (2) content similarity between events, and the (3) spatio-temporal proximity between users and events. The aim behind including the social interaction is to recommend similar events to users having a large degree of interaction, e.g., when two users mention each other frequently in their tweets. Likewise, incorporating the content similarity between events helps in recommending future events to users who have attended past events with similar topics (themes). Finally, the spatio-temporal proximity guarantees that the events that are in the vicinity of a user receive higher ranks than distant events on the same topic. In this work, we jointly account for both spatial and temporal aspects in order to avoid, for example, recommending distant events that are about to start. The primary contributions of our event recommender system are summarized as follows:

- We argue that enriching the pure model-based CF with social, topical, and spatio-temporal features helps in proving a better modeling for user preferences and in mitigating data sparsity that leads to the cold-start problem. This is because these features can be viewed as a content-based enrichment for the traditional CF techniques.

- In our model, we consider all possible associations between the two main entities, i.e., users and events, which are the user-user, event-event, and user-event associations. We also show how to regularize each of these associations with a proper contextual feature. As a result, our model is generic in the sense that

it provides a straightforward embedding and mapping of new features to their respective associations.

The rest of the chapter is organized as follows. In Section 5.2, research efforts related to event recommendation in social media are discussed. Then, we present the main notations, concepts, and problem statement in Section 5.3. In Section 5.4, the contextual features used to refine the model-based CF are addressed besides describing the steps required to integrate them into the CF model. Then, we show how to estimate the model's parameters using the gradient descent algorithm in Section 5.5. The details of our dataset and experimental evaluation are presented in Section 5.6. Finally, we conclude this chapter and discuss ongoing work in Section 5.7

5.2 Related Work

Predicting event attendance in social media has recently received a considerable attention. Apart from its vast range of applications where such systems can play a big role, the increasing interest in modeling, designing and implementing event recommender systems is mainly driven by their challenging and unique nature.

Event recommendation is closely related to predicting human mobility, which have been extensively studied in the literature. Gao and Cao [64] use the Hidden Markov Model (HMM) to characterize the behavior of user mobility. However, HMM does not take into account important factors that influence a user's choice, such as the time and location of the events and the taste of the community he/she belongs to. Asahara et al. [17] proposed a method to predict the next move of a user on the basis of the Mixed Markov Model (MMM). Their results show that their MMM-based method is substantially more accurate than MM- and HMM-based methods, because it relies not only on the user's previous status, but also on his/her personality as an unobservable parameter. Although these methods can capture human mobility, they fail to detect the irregular movements of users, i.e., event attendance. Boutsis et al. [36] introduced a MMM-based model, called PRESENT, allowing for personalized event recommendation by extracting the behavioral patterns of users in social groups. One drawback of their approach is that it confines recommended events to those that are to be attended by the members of the same social group.

Collaborative filtering is among the most popular techniques employed for event recommendation [52, 89, 117]. However, for time-specific items, e.g., events, these techniques have difficulties in dealing with events as they typically receive their ratings only after an event ends. Although some EBSNs, such as Meetup, provide

users the ability to respond to formal upcoming event invitations (RSVPs) that show which users will attend which events, a broad range of events are not covered by this new invitation procedure. To cope with that, Minkov et al. [117] proposed a model that combines both content-based and collaborative filtering and showed that collaborative predictions of future events are more effective than pure content-based recommendations. A similar hybrid content and collaborative filtering approach CF-CB for event recommendation was proposed by Cornelis et al. [52], which models users and items similarities in a fuzzy relational framework. The underlying goal of both efforts [52, 117] is recommending upcoming events if they are similar to past events that similar users have attended. Our approach follows this line of adapted collaborative filtering. However, we go further and include the user interaction as a constraint that influences the similarity between users' tastes on one hand, and the spatio-temporal features, which considerably affect the interest of users in attending events.

5.3 Preliminaries and Problem Statement

In this section, we present the notations, main concepts, and problem statement required for describing our event recommendation model.

Table 5.1: Main notations used for describing our event recommendation model.

Notation	Description
n	number of users
m	number of events
U	set of users
E	set of social events
k	dimensionality of latent factor space
$\mathbf{u}_i$	k-dimensional representation of user i
$\mathbf{e}_j$	k-dimensional representation of event j
$\mathbf{A}$	user-event attendance matrix
$ul_i(el_j)$	geo-coordinates of user i (event j)
t_j	start time of event j
T_j	a set of tags describing event j

Table 5.1 lists the notations used throughout this chapter. In Section 5.3.1, we begin by describing the main two input entities of our model: *users* and *events*. Then, the technique upon which we build our context-aware recommender system is

briefly discussed in Section 5.3.2, and finally, we present our problem statement and explicitly address the major challenges we aim to tackle.

5.3.1 User-event Attendance Representation

The main objective behind building event recommender systems is to provide users with a list of ranked events that match their preferences. Therefore, the two input entities for such systems are:

(1) a set of *users* U. Each user $u \in U$ is associated with geo-coordinates $l_u = (lat, lon)$ representing his/her current location.

(2) a set of *events* E. Each event $e \in E$ has a geo-location $l_e = (lat, lon)$, start time t_e, and a set of tags T_e reflecting the event's main theme.

In the context of event recommendation, one is commonly given a user-event attendance matrix $\mathbf{A}$ encapsulating the event attendance history of users. The temporal aspect is important since only upcoming events can be recommended to users, and hence, events are divided into *past* and *upcoming* events[1]. For past events, as can be seen in Figure 5.1, corresponding entries in $\mathbf{A}$ are either set to 1 when an event was attended by a user or to 0 otherwise. However, the entries of upcoming events[2] have no values, i.e., missing values. The goal is to estimate the values of such entries in order to predict whether a user is interested in attending an upcoming event or not. Some of these entries might be assigned 1 when a user shows his/her willing to attend an upcoming event by, for example, responding to a formal event invitation (RSVP) as done in Meetup.

New users who have recently added to the system and have no (or a very sparse) attendance history cause the so-called cold-start problem that needs special emphasis while applying collaborative filtering. This problem stands against producing reliable recommendations for such new users.

An event recommendation algorithm takes the set of users U and the set of events E as input and learns a function $\hat{\mathbf{A}}$ such that

$$\hat{\mathbf{A}} : U \times E \rightarrow \mathbb{R} \tag{5.1}$$

[1] An ongoing event is considered a future event. By this, a user is given the chance to attend this event, in particular, if it suits his/her taste and is taking place in the vicinity.

[2] The terms "future" and "upcoming" events are used interchangeably in this chapter.

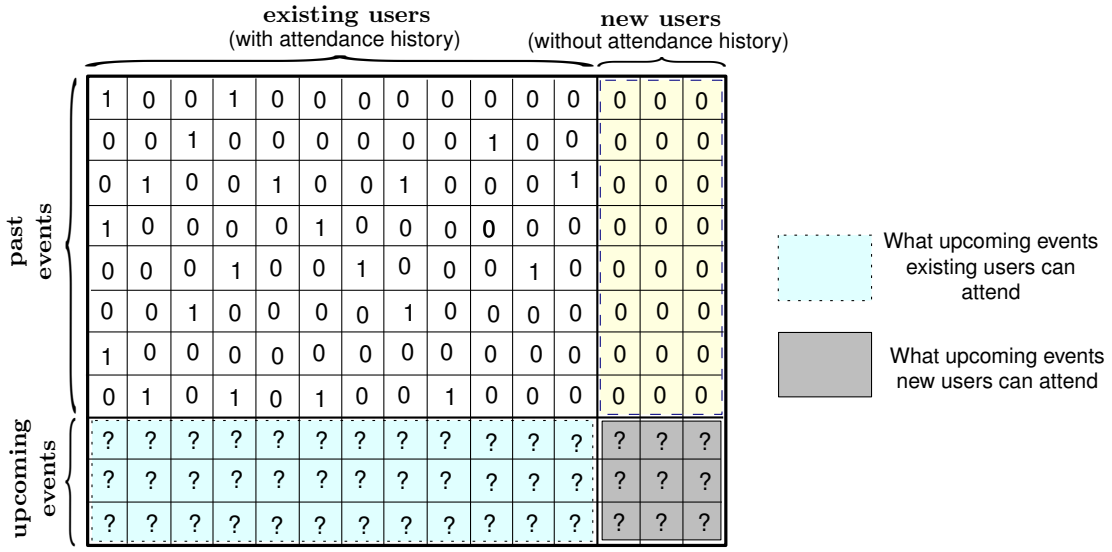


Figure 5.1: User-event Attendance matrix. For past events, an entry is set to 1 if it pertains to a user who attended the corresponding event, and to 0 otherwise. For future events, the entries have no values and need to be estimated by the model.

Using function $\hat{\mathbf{A}}$, a real value $\hat{\mathbf{A}}_u^e$ is estimated and assigned to each user-event pair (u, e), capturing the expected degree of interest of user u in attending event e. One way to realize and learn such a function is using model-based collaborative filtering, whereby the entries of matrix A are assumed to be governed by an underlying model and are not generated at random. This requires uncovering latent factors (features) that perfectly describe the interaction between users and events. Therefore, model-based collaborative filtering is sometimes referred to as *latent factor models*. The goal then is to approximate such models and to use them in estimating $\hat{\mathbf{A}}_u^e$ for each $u \in U$ and $e \in E$. These methods have recently become popular due to their good scalability and the accuracy of the results [92].

5.3.2 Regularized Singular Value Decomposition

Among a number of model-based techniques that can be employed to extract latent features, reflecting the user-event attendance patterns, we consider the method of *matrix factorization*. Loosely speaking, matrix factorization characterizes both users and events by vectors of factors inferred from the user-event attendance patterns in $\mathbf{A}$. Each vector consists of k latent factors such that $k \ll min(m, n)$, in a sense that these vectors can be viewed as k-dimensional representations of the rows and columns of matrix $\mathbf{A}$. In this reduced k-dimensional space, the noisy nature of $\mathbf{A}$ is mitigated and thus similar points becomes much closer [165]. By this, the interaction between

users and events are modeled as dot products in that reduced space. Each user u is represented by a factor vector $\mathbf{u} \in \mathbb{R}^k$, and each event e is characterized by the factor vector $\mathbf{e} \in \mathbb{R}^k$. The overall interest of user u in attending event e is then the dot product of their respective vectors, i.e.,

$$\hat{\mathbf{A}}_u^e = \mathbf{u} \cdot \mathbf{e} \tag{5.2}$$

The main task now is to estimate proper values for each vector of both users and events. For this, the well-established *Singular Value Decomposition* (SVD) is used for collaborative filtering [63]. SVD has been effectively used in estimating the latent semantic factors in the domain of natural language processing [68]. Using SVD, the latent factors are estimated by minimizing the sum of squared residuals for each user $u \in U$ and each event $e \in E$. Formally,

$$\underset{\forall u \in U, \forall e \in E}{\arg\min} \frac{1}{2} \sum_{u \in U} \sum_{e \in E} (\mathbf{A}_u^e - (\mathbf{u} \cdot \mathbf{e}))^2. \tag{5.3}$$

However, $\mathbf{A}$ is a relatively dense matrix, in particular, the matrix portion relating to past events, which results in a computationally-expensive learning procedure. To mitigate this, entries set to 0 can be ignored, i.e., treated as missing values, and directly modeling the observed entries that assigned a value of 1 [123]. To ensure that the missing values do not contribute to the summation being minimized, Eq. 5.3 is changed to

$$\underset{\forall u \in U, \forall e \in E}{\arg\min} \frac{1}{2} \sum_{u \in U} \sum_{e \in E} I^e (\mathbf{A}_u^e - (\mathbf{u} \cdot \mathbf{e}))^2 + \frac{\lambda}{2} \sum_{u \in U} \|\mathbf{u}\|^2 + \frac{\lambda}{2} \sum_{e \in E} \|\mathbf{e}\|^2, \tag{5.4}$$

where

$$I^e := \begin{cases} 0 & \text{if } (e \text{ is an upcoming event}) \\ 1 & \textit{otherwise} \end{cases} \tag{5.5}$$

Consequently, the recommendation algorithm learns the model parameters by fitting the observed entries of past events, resulting in accurately estimating these observed patterns. However, the algorithm might fail to estimate the attendance patterns for missing values because of the *overfitting* problem. To avoid this, the learned factors are regularized using the terms $(\frac{\lambda}{2} \sum_{u \in U} \|\mathbf{u}\|^2)$ and $(\frac{\lambda}{2} \sum_{e \in E} \|\mathbf{e}\|^2)$, where $\|\mathbf{u}\|$ and

$\|\mathbf{e}\|$ refers to the 2-norm of $\mathbf{u}$ and $\mathbf{e}$, respectively. The 2-norm of $\mathbf{u}$ is calculated as

$$\|\mathbf{u}\| = \sqrt{\sum_{i=1}^{k} \mathbf{u}_i^2}, \tag{5.6}$$

where $\mathbf{u}_i$ is the i-th factor of vector $\mathbf{u}$. The magnitudes of the regularized factors are penalized so that only important factors are considered to represent the observations, which is achieved by having small vector norms for both $\mathbf{u}$ and $\mathbf{e}$. Therefore, these models are referred to as *Regularized SVD* [123]. The parameter λ is a predetermined constant that controls the impact of the regularization terms on the optimization process. A small λ might not help solving the overfitting problem, while a large value of λ leads to reduce the influence of the activity history of users in directing the ranking process. In Section 5.6, we show how to set the value of λ using a cross-validation-like method.

5.3.3 Problem Statement

The problem of event recommendation can be stated as follows. Given a certain user u, the main task is *to provide u with a ranked list of future events E_u that match his/her preferences*. The traditional CF fails to accurately estimate this ranked list because of (1) the sparse nature of the attendance matrix $\mathbf{A}$, which leads to the cold-start problem and (2) its inability to capture contextual aspects, e.g., social links and the topics of events, which are are essential for a more reliable recommendation. Therefore, the major objective of this study is *to define a set of features, namely, social, topical, and spatio-temporal features*, that can be viewed as content-based enrichment to CF. Moreover, we will show how *to integrate these features into the regularized SVD model*.

5.4 Context-aware Event Recommendation

In this section, we describe our context-aware event recommender system. First, in Section 5.4.1, the proposed contextual features are listed and detailed. Then, in Section 5.4.2, we show how to include these features in the matrix factorization model towards building a context-aware recommender system.

5.4.1 Contextual Features

Three types of features including *social*, *topical*, and *spatio-temporal* features are proposed and used to refine the traditional model-based collaborative filtering.

Social feature

People prefer to attend events that their close friends, relatives, or colleagues are willing to attend. Therefore, the interaction between users, i.e., social links, is an important factor for event recommender systems [165]. Quantifying and including this interaction as an additional feature to these systems helps direct the learning process so that the taste of user u, which is encoded in vector $\mathbf{u}$, is affected more by the taste of the users he/she interacts with.

We model the user interaction as an undirected graph where each node represents exactly one user. An edge is established between two nodes if and only if a certain relationship is recognized between corresponding users. The edges are weighted and each weight reflects the degree of interaction between users.

Example 5.1 *In Meetup, the weight of an edge between two users might refer to the number of groups they join in common. However, in Twitter, this weight might reflect the number of times they mention each other during a time window.*

Let $w_{i,j}$ denote the weight of the edge linking user u_i and user u_j. Then, the interaction degree between both users is defined as

Definition 5.1 (Interaction Degree) *The interaction degree $\mathcal{I}_{i,j}$ between user u_i and user u_j is the ratio between the weight $w_{i,j}$ and the sum of the weights of the edges connecting node i with its direct neighbors $\mathcal{A}_i$, i.e,*

$$\mathcal{I}_{i,j} := \frac{w_{i,j}}{\sum_{a \in \mathcal{A}_i} w_{i,a}}. \tag{5.7}$$

Note that $\mathcal{I}_{i,j}$ is not necessarily equal to $\mathcal{I}_{j,i}$ because the users u_i and u_j might have different neighbors, i.e., $\mathcal{A}_j \neq \mathcal{A}_i$.

Topical feature

This feature captures the semantic similarity between events. The aim is to identify similar events, to quantify this similarity, and then, to direct the learning process so that the distance between the vectors of on-topic events is reduced. As a result, the recommender system will favor the events that are semantically-compatible with what

a user has already attended over other events. To estimate the thematic similarity between the events e_i and e_j, we use Jaccard similarity coefficient [126], i.e.,

$$\mathcal{T}_{i,j} = \frac{|T_i \bigcap T_j|}{|T_i \bigcup T_j|}, \tag{5.8}$$

where T_i and T_j are the set of tags, e.g., uni-grams and bi-grams, that are associated with events e_i and e_j, respectively. We choose to use this simple text similarity measure based on the assumption that tags attached to each event are commonly encompassing a list of preprocessed and representative keywords with less noise.

Spatio-temporal features

To provide a spatio-temporal-aware system that better suits the preference of users, we incorporate spatial and temporal features into our context-aware event recommender system. In the following, we first show how to extract the spatial feature f_s and temporal feature f_t, which are then combined as a single spatio-temporal feature $\mathcal{S}_u^e$ that reflects the interaction between both dimensions.

Spatial feature. The spatial proximity between the current location l_u of user u and the location l_e of event e is a key factor that affects the attendance patterns. That is, the closer u is to the location of event e, the more likely it is that u will attend event e. To model this spatial feature, we choose to apply the Gaussian function:

$$f_s(d_u^e) = a\ e^{-\frac{(d_u^e - b)^2}{2\sigma^2}}, \tag{5.9}$$

where d_u^e is the Haversine [141] distance between l_u and l_e, and the constants a, b, σ refer to the height of the peak of the function's curve, the point at which f_s is maximized (mode), and the standard deviation that controls the width of the curve, respectively. This function is monotonically decreasing and therefore assigns higher weights to closer distances between users and events. To get a maximum value of 1 when $d_u^e = 0$, we set a and b to 1 and 0, respectively. Therefore, the spatial feature $f_s(d_u^e)$ capturing the likelihood that user u attends event e is given by

$$f_s(d_u^e) = e^{-\frac{(d_u^e)^2}{2\sigma^2}}. \tag{5.10}$$

We tune the parameter σ using cross-validation as will be discussed in Section 5.6.

Temporal feature. Users usually need enough time to make a decision on whether to attend an event so that their current plans are not affected. For this, the likelihood that a user will attend an event can be estimated based on the temporal gap (measured in hours) between the start time t_e of e and the current time t. In case that a user did not know about an event a priori, but shortly before it starts, the chance that he/she will attend it is in general low. The temporal feature is estimated based on this premise using the Sigmoid function defined as

$$f_t(\Delta t_e) = \frac{1}{1 + e^{-s\Delta t_e}}. \tag{5.11}$$

where $\Delta t_e = t_e - t$ is the time left for event e to start. The larger the temporal gap Δt_e, the closer the returned value to 1 is. To force this function to return a value in the range $[0, 1]$, two translation operations are applied to this function: 1) moving the function down 0.5 unit, 2) stretching the function by multiplying it by 2. Formally,

$$f_t(\Delta t_e) = 2\left(\frac{1}{1 + e^{-s\Delta t_e}} - 0.5\right). \tag{5.12}$$

Therefore, the temporal feature of event e is defined as

$$f_t(\Delta t_e) = \frac{1 - e^{-s\Delta t_e}}{1 + e^{-s\Delta t_e}} \tag{5.13}$$

where the parameter s controls the steepness of the curve, i.e., how fast the returned value approaches 1 as Δt_e increases. Heuristically, we set s to 0.03 so that the temporal feature becomes close to 1 when Δt_e is about one week, as illustrated in Figure 5.2.

Finally, we define the spatio-temporal feature $\mathcal{S}_u^e$ as a linear combination of both the spatial $f_s(d_u^e)$ and temporal $f_t(\Delta t)$ features as follows.

$$\mathcal{S}_u^e := 0.5 f_s(d_u^e) + 0.5 f_t(\Delta t_e), \tag{5.14}$$

This spatio-temporal interaction, defined in Eq. 5.14, helps avoid underestimating or overestimating the preference of a user, which occurs when each dimension is handled separately. For example, the value of the spatial feature $f_s(d_u^e) \approx 0$ when the location of e is too far from the location of user u. However, user u might still be interested in attending e if he/she has enough time to arrange for the attendance, which is realized by the temporal feature $f_t(\Delta t_e) \approx 1$. As a result, in this particular

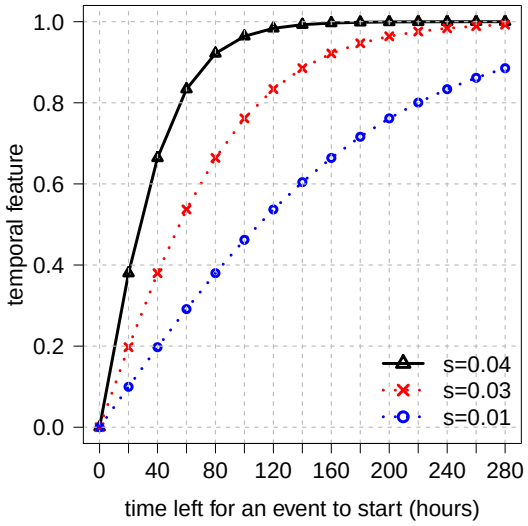


Figure 5.2: Impact of varying the parameter s in Eq. 5.13 on the temporal feature.

scenario, the spatio-temporal feature describing the likelihood that user u will attend this spatially-far event e is $\mathcal{S}_u^e \approx 0.5$ instead of 0.

5.4.2 Context-aware SVD model

In Section 5.3.2, we described the regularized SVD model that is used to estimate the k-dimensional factor vectors $\mathbf{u}$ ($\forall u \in U$) and $\mathbf{e}$ ($\forall e \in E$). These vectors are then utilized to approximate the probability that user u will attend event e by multiplying both vectors $\mathbf{u}$ and $\mathbf{e}$ as shown in Eq. 5.2. In this section, the contextual features addressed in Section 5.4.1 are employed to adapt the regularized SVD, making it social-, topical-, and spatio-temporal-aware model. In the following, we discuss how to integrate these contextual features into the regularized SVD model.

When two users u_i, u_j have a high-degree of interaction (a high value of $\mathcal{I}_{i,j}$), the similarity between their respective factor vectors $\mathbf{u}_i, \mathbf{u}_j$ needs to be maximized. In mathematical terms, this is equivalent to maximizing the term

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \mathcal{I}_{i,j} \times cosine(\mathbf{u}_i, \mathbf{u}_j), \tag{5.15}$$

where $cosine(\mathbf{u}_i, \mathbf{u}_j)$ is the cosine similarity between the vectors u_i, u_j, which is given as

$$cosine(\mathbf{u}_i, \mathbf{u}_j) = \frac{\mathbf{u}_i \cdot \mathbf{u}_j}{\|\mathbf{u}_i\| \times \|\mathbf{u}_j\|}. \tag{5.16}$$

Likewise, the similarity between two events that share some keywords, i.e., on-topic events, needs to be emphasized by maximizing the similarity between their respective vectors. This is accomplished by maximizing the term

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\mathcal{T}_{i,j} \times cosine(\mathbf{e}_i, \mathbf{e}_j). \tag{5.17}$$

As for including the spatio-temporal feature into the regularized SVD, we need to do the following:

(1) *Imputation*: Since we are only interested in recommending upcoming events to users, the entries in matrix $\mathbf{A}$, corresponding to such events, are set to 1. Therefore, the entries of $\mathbf{A}$ are reset as follows.

$$\mathbf{A}_i^j := \begin{cases} 1 & \text{if } t_j > t \\ \mathbf{A}_i^j & otherwise \end{cases} \tag{5.18}$$

(2) *Spatio-temporal regularization*: The imputation in the first step implies that users are interested in attending all future events, which is, of course, inaccurate. Therefore, the spatio-temporal feature is used to mitigate the impact of the imputation by regularizing the user-event interaction. That is, the role of the spatio-temporal feature $\mathcal{S}_i^j$ is to control the influence of minimizing the squared residual in case of user u_i and event e_j. By applying this spatio-temporal regularization, the sum of squared residuals term in Eq. 5.4 becomes

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\mathcal{S}_i^j I^j(\mathbf{A}_i^j - (\mathbf{u}_i \cdot \mathbf{e}_j))^2. \tag{5.19}$$

To ignore this type of regularization for past events, we reset their respective spatio-temporal features to 1, i.e.,

$$\mathcal{S}_i^j := \begin{cases} 1 & \text{if } t_j < t \\ \mathcal{S}_i^j & otherwise \end{cases} \tag{5.20}$$

where t is the current time. We call the terms in Eq. 5.15, Eq. 5.17, and Eq. 5.19 the social, topical, and spatio-temporal constraints, respectively, which can be introduced as new additive terms in Eq. 5.4. Therefore, our final goal is to minimize the cost

function $\mathcal{L}(\mathbf{U}, \mathbf{E})$ defined as:

$$\mathcal{L}(\mathbf{U}, \mathbf{E}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathcal{S}_i^j I^j(\mathbf{A}_i^j - (\mathbf{u}_i \cdot \mathbf{e}_j))^2 - \theta\sum_{i=1}^{n}\sum_{j=1}^{n}\mathcal{I}_{i,j} \times cosine(\mathbf{u}_i, \mathbf{u}_j) - \beta\sum_{i=1}^{m}\sum_{j=1}^{m}\mathcal{T}_{i,j} \times cosine(\mathbf{e}_i, \mathbf{e}_j) + \frac{\lambda}{2}\sum_{i=1}^{n}\|\mathbf{u}_i\|^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\|\mathbf{e}_i\|^2, \quad (5.21)$$

where the parameters θ and β are the social and topical coefficients, respectively. They control the influence of their respective constraints. The negatives of both the social and topical constraints are used because maximizing a certain function is equivalent to minimizing the same function but with a sign change. Finding a local minimum for this optimization problem can be achieved using gradient-descent, which is used to approximate the vectors $\mathbf{u}$ ($\forall u \in U$) and $\mathbf{e}$ ($\forall e \in E$), as discussed in the following section.

It is noteworthy that our model as can be inferred from Eq. 5.21 covers all possible interactions between the system's main entities: users and events. Therefore, one can easily include new features by updating one (or a combination) of the features $\mathcal{I}_{i,j}$, $\mathcal{T}_{i,j}$, and $\mathcal{S}_i^j$. For example, if one needs to allow the recommender system to incorporate synonyms into the learning process, only the topical feature $\mathcal{T}_{i,j}$ is updated without any further modifications.

5.5 Learning Algorithm

An important step to figure out whether user i is interested in attending event j is to estimate the factors (parameters) of their respective vectors $\mathbf{u}_i$ and $\mathbf{e}_j$. Assume that the set of factor vectors is denoted $\mathbf{U} \in \mathbb{R}^{n\times k}$ and $\mathbf{E} \in \mathbb{R}^{m\times k}$ such that $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n\}$ and $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m\}$ for users and events, respectively. Estimating the factor vectors $\mathbf{U}$ and $\mathbf{E}$ requires minimizing the cost function given in Eq. 5.21. One of the well-known algorithms used for minimizing such a function is *gradient descent* [144].

In Section 5.5.1, we describe the basic idea behind gradient descent and how to apply it for our problem settings. Then, we detail the steps needed to derive and estimate the *gradients* that are used to direct the convergence of the learning algorithm in Section 5.5.2.

5.5.1 Gradient Descent Algorithm

Gradient descent (GD) algorithm is a search algorithm that is mainly used for function minimization. To minimize our cost function $\mathcal{L}(\mathbf{U}, \mathbf{E})$ using gradient descent, the parameters[1] $\mathbf{U}, \mathbf{E}$ are randomly-initialized and GD moves toward a set of parameter values that minimize $\mathcal{L}(\mathbf{U}, \mathbf{E})$. In each iteration of GD, the algorithm takes a step in the negative direction of the *gradient*, i.e., a step in the direction of the steepest decrease of $\mathcal{L}(\mathbf{U}, \mathbf{E})$, until convergence. Algorithm 5.1 lists the main steps required to update the parameters using GD.

Algorithm 5.1: Main steps of Gradient Descent.

Input: user vectors $\mathbf{U}$ and event vectors $\mathbf{E}$
Output: updated $\mathbf{U}$ and $\mathbf{E}$, which minimizes $\mathcal{L}$

initialize $\mathbf{U}$ *and* $\mathbf{E}$ *randomly*
while *not converged* **do**
    `// Updating user vectors`
    for $r = 1$ *to* n **do**
        for $p = 1$ *to* k **do**
            $\mathbf{u}_r^p \leftarrow \mathbf{u}_r^p - \alpha \frac{\partial}{\partial \mathbf{u}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$
    `// Updating event vectors`
    for $r = 1$ *to* m **do**
        for $p = 1$ *to* k **do**
            $\mathbf{e}_r^p \leftarrow \mathbf{e}_r^p - \alpha \frac{\partial}{\partial \mathbf{e}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$
return $\mathbf{U}, \mathbf{E}$

The parameters $\mathbf{U}$ and $\mathbf{E}$ are updated simultaneously so that updating one parameter does not affect the values of other parameters within the same iteration. The parameter α is called the *learning rate* and is used to control how fast the algorithm converges to a stationary point. Too small values of α result in a slow convergence, while high values might lead to an undesirable divergence. The terms $\frac{\partial}{\partial \mathbf{u}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$ and $\frac{\partial}{\partial \mathbf{e}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$ are called the gradients of the function, and their values are iteratively used to update the user and event vectors, respectively. In the following section, we present how to estimate these gradients.

[1] The notion "parameters" refers to the factors of the vectors of both $\mathbf{U}, \mathbf{E}$

5.5.2 Gradient Estimation

The gradient $\frac{\partial}{\partial \mathbf{u}_r^p}\mathcal{L}(\mathbf{U},\mathbf{E})$ of the function at $\mathbf{u}_r^p$, which is the partial derivate of $\mathcal{L}$ with respect to $\mathbf{u}_r^p$, is given as

$$\begin{aligned}\frac{\partial}{\partial \mathbf{u}_r^p}\mathcal{L}(\mathbf{U},\mathbf{E}) = &\underbrace{\frac{\partial}{\partial \mathbf{u}_r^p}\left(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathcal{S}_i^j I_j(\mathbf{A}_i^j - (\mathbf{u}_i\cdot\mathbf{e}_j))^2\right) + \frac{\partial}{\partial \mathbf{u}_r^p}\left(\frac{\lambda}{2}\sum_{i=1}^{n}\|\mathbf{u}_i\|^2\right)}_{T_1}\\ &\underbrace{-\frac{\partial}{\partial \mathbf{u}_r^p}\left(\theta\sum_{i=1}^{n}\sum_{j=1}^{n}\mathcal{I}_{i,j}\times cosine(\mathbf{u}_i,\mathbf{u}_j)\right)}_{T_2},\end{aligned} \tag{5.22}$$

In Eq. 5.22, Term T_1 refers to the partial derivative of both the sum of squared residuals and the user regularization term with respect to $\mathbf{u}_r^p$ and can be reduced to[1]

$$T_1 = \left(\sum_{j=1}^{m}\mathcal{S}_r^j I_j(\mathbf{A}_r^j - (\mathbf{u}_r\cdot\mathbf{e}_j))\,\mathbf{e}_j^p\right) + \lambda\mathbf{u}_r^p, \tag{5.23}$$

where the squared residuals are affected more by larger spatio-temporal features $\mathcal{S}_r^j$. This enables the model to avoid recommending events to users for low values of $\mathcal{S}_r^j$, e.g., when the user is located far away from the event and the event is occurring soon.

Term T_2 corresponds to the partial derivative of the social constraint. Substituting $cosine(\mathbf{u}_i,\mathbf{u}_j)$ in term T_2 by the result of Eq. 5.16, term T_2 becomes

$$\begin{aligned}T_2 =&\frac{\partial}{\partial \mathbf{u}_r^p}\left(\theta\sum_{i=1}^{n}\sum_{j=1}^{n}\mathcal{I}_{i,j}\frac{\mathbf{u}_i\cdot\mathbf{u}_j}{\|\mathbf{u}_i\|\times\|\mathbf{u}_j\|}\right)\\ =&\theta\sum_{j=1}^{n}\mathcal{I}_{r,j}\frac{\partial}{\partial \mathbf{u}_r^p}\left(\frac{\mathbf{u}_r\cdot\mathbf{u}_j}{\|\mathbf{u}_r\|\times\|\mathbf{u}_j\|}\right) + \theta\sum_{i=1}^{n}\mathcal{I}_{i,r}\frac{\partial}{\partial \mathbf{u}_r^p}\left(\frac{\mathbf{u}_i\cdot\mathbf{u}_r}{\|\mathbf{u}_i\|\times\|\mathbf{u}_r\|}\right).\end{aligned} \tag{5.24}$$

Since both summations in Eq. 5.24 sums over the same items (users), and according to the commutative law, we restructure the equation as follows:

$$T_2 = \theta\sum_{i=1}^{n}(\mathcal{I}_{i,r} + \mathcal{I}_{r,i})\underbrace{\frac{\partial}{\partial \mathbf{u}_r^p}\left(\frac{\mathbf{u}_i\cdot\mathbf{u}_r}{\|\mathbf{u}_i\|\times\|\mathbf{u}_r\|}\right)}_{T_3}. \tag{5.25}$$

[1]Note that the partial derivative of both the topical constraint and the event regularization term are 0, and hence, are omitted from Eq. 5.22.

Using calculus, term T_3 can be calculated as follows:

$$\begin{aligned} T_3 =& (\mathbf{u}_i \cdot \mathbf{u}_r) \times \frac{\partial}{\partial \mathbf{u}_r^p} \left(\frac{1}{\|\mathbf{u}_i\| \times \|\mathbf{u}_r\|} \right) + \frac{1}{\|\mathbf{u}_i\| \times \|\mathbf{u}_r\|} \times \frac{\partial}{\partial \mathbf{u}_r^p} (\mathbf{u}_i \cdot \mathbf{u}_r) \\ =& - \frac{(\mathbf{u}_i \cdot \mathbf{u}_r) \times \mathbf{u}_r^p}{\sqrt{\|\mathbf{u}_r\| \times \|\mathbf{u}_i\|^3}} + \frac{\mathbf{u}_i^p}{\|\mathbf{u}_i\| \times \|\mathbf{u}_r\|} \end{aligned} \tag{5.26}$$

Finally, by applying the result of Eq. 5.26 to its corresponding term in Eq. 5.25 and by substituting T_1 and T_2 in Eq. 5.22 by the results of Eq. 5.23 and Eq. 5.25, the gradient $\frac{\partial}{\partial \mathbf{u}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$ of the function at $\mathbf{u}_r^p$ is estimated as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{u}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E}) =& \left(\sum_{j=1}^{m} \mathcal{S}_r^j I_j (\mathbf{A}_r^j - (\mathbf{u}_r \cdot \mathbf{e}_j))\, \mathbf{e}_j^p \right) + \lambda \mathbf{u}_r^p \\ & - \theta \sum_{i=1}^{n} \left((\mathcal{I}_{i,r} + \mathcal{I}_{r,i}) \left[\frac{(\mathbf{u}_i \cdot \mathbf{u}_r) \times \mathbf{u}_r^p}{\sqrt{\|\mathbf{u}_r\| \times \|\mathbf{u}_i\|^3}} + \frac{\mathbf{u}_i^p}{\|\mathbf{u}_i\| \times \|\mathbf{u}_r\|} \right] \right) \end{aligned} \tag{5.27}$$

To update the event parameters $\mathbf{E}$, the partial derivative $\frac{\partial}{\partial \mathbf{e}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$ needs to be computed. It is obtained by

$$\begin{aligned} \frac{\partial}{\partial \mathbf{e}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E}) =& \underbrace{\frac{\partial}{\partial \mathbf{e}_r^p} \left(\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} \mathcal{S}_i^j I_j (\mathbf{A}_i^j - (\mathbf{u}_i \cdot \mathbf{e}_j))^2 \right) + \frac{\partial}{\partial \mathbf{e}_r^p} \left(\frac{\lambda}{2} \sum_{i=1}^{m} \|\mathbf{e}_i\|^2 \right)}_{T_1} \\ & \underbrace{- \frac{\partial}{\partial \mathbf{e}_r^m} \left(\beta \sum_{i=1}^{m} \sum_{j=1}^{m} \mathcal{T}_{i,j} \times cosine(\mathbf{e}_i, \mathbf{e}_j) \right)}_{T_2} \end{aligned} \tag{5.28}$$

Term 2 in Eq 5.28 refers to the topical constraint that contributes to reducing the distance between event vectors when they share similar tags, i.e., on-topic events. Similar to the simplification steps that are applied to Eq. 5.22, the gradient $\frac{\partial}{\partial \mathbf{e}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$ of the function at $\mathbf{e}_r^p$ can be rewritten as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{e}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E}) =& \left(\sum_{i=1}^{n} \mathcal{S}_i^r I_r (\mathbf{A}_i^r - (\mathbf{u}_i \cdot \mathbf{e}_r))\, \mathbf{e}_i^p \right) + \lambda \mathbf{e}_r^p \\ & - \beta \sum_{i=1}^{m} \left(\mathcal{T}_{i,r} \left[\frac{(\mathbf{e}_i \cdot \mathbf{e}_r) \times \mathbf{e}_r^p}{\sqrt{\|\mathbf{e}_r\| \times \|\mathbf{e}_i\|^3}} + \frac{\mathbf{e}_i^p}{\|\mathbf{e}_i\| \times \|\mathbf{e}_r\|} \right] \right) \end{aligned} \tag{5.29}$$

As a result, we have an estimated value for both gradients $\frac{\partial}{\partial \mathbf{u}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$ and $\frac{\partial}{\partial \mathbf{e}_r^p} \mathcal{L}(\mathbf{U}, \mathbf{E})$, which are used to update the values of the factors $\mathbf{u}_r^p$ and $\mathbf{e}_r^p$, respec-

tively, as illustrated in Algorithm 5.1. The factors of both $\mathbf{U}$ and $\mathbf{E}$ are iteratively updated until convergence.

Finally, it is noteworthy that we have implemented this model completely since, to the best of our knowledge, there is no existing SVD-based learning package that enables the inclusion of additional features similar to those proposed in this work.

Runtime Complexity

We discuss now the runtime complexity of our model with n users, m events, and k-dimensional vectors. Since the regularized version of SVD is employed, we do not iterate over all possible permutations of the model's entities. For example, Eq. 5.23 can be rewritten as

$$T_1 = \left(\sum_{j \in E_u} \mathcal{S}_r^j (\mathbf{A}_r^j - (\mathbf{u}_r \cdot \mathbf{e}_j))\, \mathbf{e}_j^p \right) + \lambda \mathbf{u}_r^p. \tag{5.30}$$

That is, to compute T_1, we need $|E_u|$ iterations instead of m, where E_u is the set of events that u has attended. Of course, $|E_u|$ is much smaller than m. This is also valid when computing the topical and social constraints. For example, when computing T_2 in Eq. 5.25 for user u_r, we need to iterate over only the users who interact with u_r. Based on this, the following statistics are needed to compute the time complexity:

- ϵ: the number of pairs of users interacting with each other. In fact, $\epsilon \ll \frac{n^2-n}{2}$ because a user generally interacts with a small subset of U.
- ρ: the number of pairs of events that share at least one tag in common. Likewise, $\rho \ll \frac{m^2-m}{2}$ due to the heterogeneity of topics and the assumed informative nature of tags describing each event.
- μ_U: the average number of users per event.
- μ_E: the average number of events per user.

At each iteration, the following statistics required for subsequent steps are estimated and cached:

(1) user and event norms, i.e., $\|\mathbf{u}\|$ and $\|\mathbf{e}\|$. This requires nk and mk operations for all users and events, respectively.

(2) the dot product of all interacting users, which needs $nk\epsilon$ operations.

(3) the dot product of all events having at least one tag in common. This step requires $mk\rho$ operations.

(4) the dot product $(\mathbf{u} \cdot \mathbf{e})$ for all users and only the events they have attended, requiring $n\mu_E k$ operations.

To compute the gradient $\frac{\partial}{\partial \mathbf{u}_r^p}\mathcal{L}(\mathbf{U}, \mathbf{E})$, about $(\mu_E + \epsilon)$ operations are required. Hence, the total number of operations performed to update the k factors of each user $u \in U$ is roughly $nk(\mu_E + \epsilon)$. As for computing the gradient $\frac{\partial}{\partial \mathbf{e}_r^p}\mathcal{L}(\mathbf{U}, \mathbf{E})$, approximately $(\mu_U + \rho)$ operations are performed, which results in $mk(\mu_U + \rho)$ operations to update the k factors of each event $e \in E$. Therefore, the total number of required operations per iteration is about $nk + mk + nk\epsilon + mk\rho + nk\mu_E + nk(\mu_E + \epsilon) + mk(\mu_U + \rho)$ operations. As a result, the runtime complexity of our proposed model is

$$O(nk(\mu_E + \epsilon) + mk(\mu_U + \rho))$$

Updating the vectors $\mathbf{U}$ and $\mathbf{E}$, as illustrated in Algorithm 5.1 (Lines 3-7), requires computing the gradients of all users and events, which is a computationally-expensive operation. To speedup computation, we make use of parallel processing by splitting the set of users and events among multiple processing cores (threads) when updating the user and event vectors, respectively.

5.6 Experimental Evaluation

To evaluate the performance of our model, we first present our dataset, the employed evaluation metrics, and the parameter settings in Section 5.6.1. Then, in Section 5.6.2, we report the results and compare our model with a number of baseline methods including the traditional CF.

5.6.1 Experimental Settings

Dataset

The dataset used in the evaluation was obtained from the popular Meetup social network from October 2011 to January 2012, which is publically available over the Internet [107]. This dataset includes a large and sparse attendance matrix, where one can find a large number of users who attended only one event that has not been attended by someone else. To mitigate this sparsity, we selected the users who

attended at least 8 events, resulting in a total of 10511 users and 2269 events, where about 0.6% of the attendance matrix's entries hold 1s. The users are distributed in groups, where each user can register in several groups. These groups are viewed as communities of users, reflecting the user interaction considered in this study as an important context-aware feature. All the data are anonymized, so that the user, group, and event ids in this dataset have no one-to-one mapping to real ids in Meetup.

The total number of distinct tags retrieved from this dataset is 1010 tags. In fact, the crawled events are not directly associated with tags. However, the crawled data contain two types of associations that can help in enriching events with tags. More specifically, the dataset provides both user-group and group-tag pairs. For each event, we first identified the groups that contain at least one user who attended the event, and then assigned the intersection of their tags to that event, based on the premise that the tags of each group is a good semantic dimension reflecting the overall interest of users belonging to that group.

Evaluation Methods

To conduct an overall evaluation of the effectiveness of our proposed model, we first present the following three recommendation scenarios:

(1) New Users NU: To test the ability of the system to recommend events to new users having no attendance history, we randomly select 10% of the users, reset their corresponding entries in $\mathbf{A}$ to 0, and kept the original values as ground truth. The dataset containing only the selected users is referred to as NUh, while the entire dataset is denoted NUd.

(2) New Events NE: We aim here at evaluating the performance of the system in suggesting future events to the users. About 16% of the events are randomly selected as future events. We reset their entries to 0 and kept the original values as ground truth. Therefore, the dataset by this scenario is denoted NEh and NEd for the selected entries and the entire data, respectively.

(3) New Users and Events NUE: The most difficult scenario is to recommend future events to new users. To evaluate our model with respect to this scenario, we reset the users and events selected in the aforementioned two scenarios to 0 and kept the original values as ground truth. Likewise, we call the entire dataset NUEd and the selected dataset NUEh.

The model is evaluated in terms of its ability (1) to recover the attendance history and (2) to rank events based on relevancy.

Recovery of attendance history. The performance of the system in recovering the entries of the attendance matrix $\mathbf{A}$, in particular, the hidden entries as specified in NU, NE, and NUE scenarios, is tested. This proves that the system can handle the cold-start problem well. For this, we normalize the estimated predictions after fitting the model's parameters in order to have a value in the range $[0, 1]$. That is, the predicted interest $(\mathbf{u}_i \cdot \mathbf{e}_j)$ of user i in attending event j is normalized as follows:

$$N_i^j := \frac{\mathbf{u}_i \cdot \mathbf{e}_j - min}{max - min}, \tag{5.31}$$

where $min = min\{\mathbf{u}_i \cdot \mathbf{e}_j | \forall i \in U, \forall j \in E\}$ and $max = max\{\mathbf{u}_i \cdot \mathbf{e}_j | \forall i \in U, \forall j \in E\}$. By this, our model is treated as a binary classifier that returns a value in the range $[0, 1]$ for each entry in $\mathbf{A}$. Now, by choosing a cutoff threshold in that range, the confusion matrix [111] is built as shown in Table 5.2.

Table 5.2: General confusion matrix.

		Predicted class		
		true	false	Total
Actual class	true	TP	FN	P
	false	FP	TN	N
	Total	P′	N′	P+N

The confusion matrix [73], also called "contingency table", is a useful tool for studying how well a classifier can recognize records of different classes. The entries TP and TN indicate that the system is correctly classifying records, while FP and FN show cases when the records are wrongly annotated. Based on that, we use the following measures to evaluate the ability of our model in recovering entries in $\mathbf{A}$.

- **AUC** (Area Under ROC Curve). It is a statistically-consistent and discriminating metric that is tailored for evaluating the performance of binary classifiers [83]. The ROC curve plots the true positive rate TPR versus the false positive rate FPR at different threshold settings. The value of TPR, also called "recall" or "sensitivity", is given as

$$TPR = \frac{TP}{TP + FN}, \tag{5.32}$$

while the FPR, also coined "fall-out", is defined as

$$FPR = \frac{FP}{FP + TN} \tag{5.33}$$

The closer the curve is to the upper-left corner, the larger is the area under curve ROC and the better the classifier.

- **Accuracy**: It reflects how well the model specifies the class of entries in $\mathbf{A}$ and is defined as the percentage of entries that are correctly classified by the model, i.e.,

$$accuracy = \frac{TP + TN}{P + N}. \tag{5.34}$$

In fact, the entries in $\mathbf{A}$ are *class-imbalanced*, meaning that the majority of these entries represent the negative class. Therefore, the ROC curve is more appropriate than *accuracy* in testing the performance of classifiers in such a case [73]. However, we keep utilizing the *accuracy* measure to capture the relative differences between alternative methods.

Relevancy-oriented ranking. Here, we go beyond evaluating the performance of our system in recovering the hidden entries, yet study the matching degree between the top-k recommended events and the users' preferences. That is, we aim at figuring out whether the top-k ranked future events E_i^k recommended to user i match his/her taste. In fact, it is difficult to acquire a ground truth to perform such a test, and thus, we rely on the following heuristics: *if at least one user who interacts with user i has attended the event recommended to i, then this event is assumed to be relevant to i*. For this, we examine the top-k events E_i^k recommended to user i, and calculate the percentage of these events that at least one of his/her group's members is going to attend. By this, an overall precision $prec@k$ is computed for all users as follows

$$prec@k := \frac{\sum_{i \in U} \sum_{f \in E_i^k} I(\exists j \in U : (\mathcal{I}_{i,j} > 0 \ \wedge \ \mathbf{A}_j^f = 1))}{n \times k}, \tag{5.35}$$

where $I(.)$ is the identity function.

Parameters Setting

Tuning model parameters is critical to its performance. As for the dimensionality of the factor space k and the learning rate α, we follow existing work [167] and set them

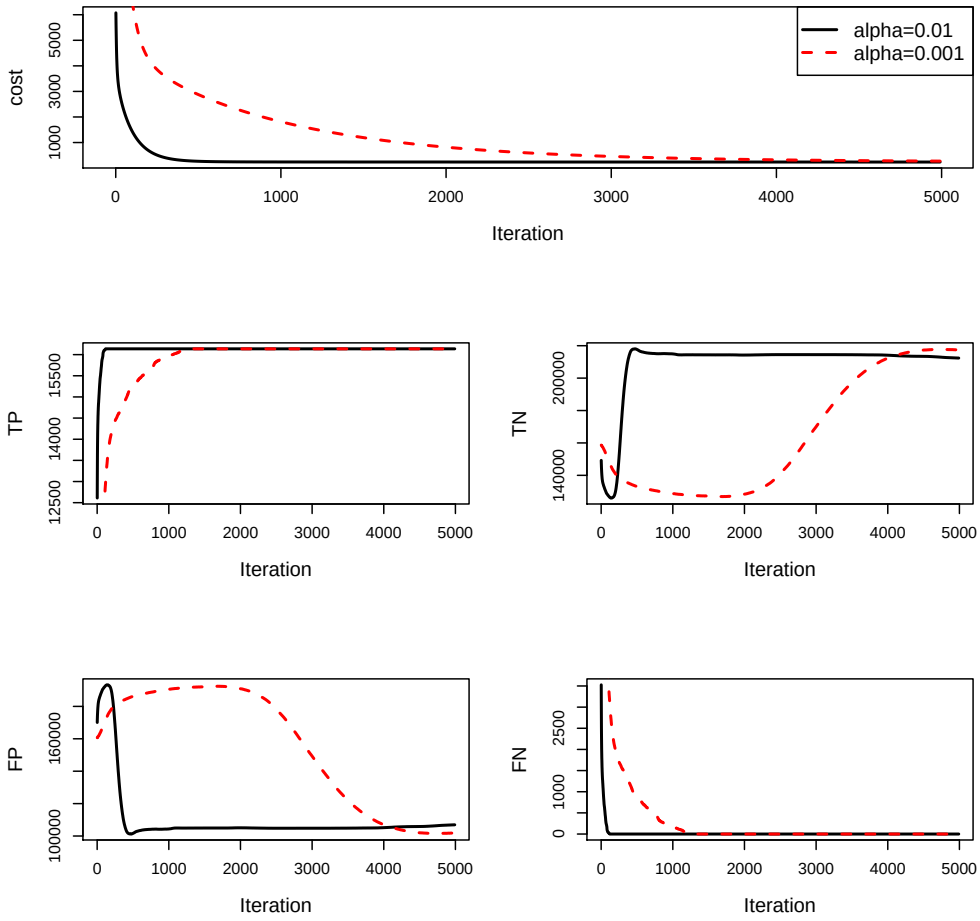
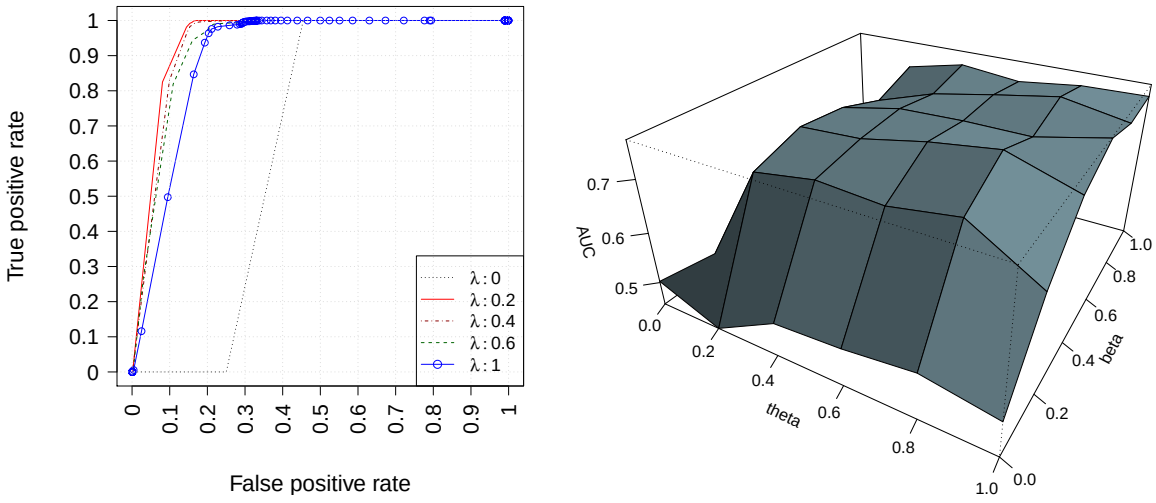


Figure 5.3: The impact of varying the learning parameters α on the convergence of the model.

to 20, 0.001, respectively. Figure 5.3 shows that choosing smaller values for α leads to a slower convergence, but closer to the local minimum.

In order to tune the regularization constant λ, the social coefficient θ, and the topical coefficient β, we conduct a cross-validation-like method. That is, we randomly choose 1% of the users as a hold-out set and set their respective entries in $\mathbf{A}$ to 0. Then, we evaluate the system's performance on a certain parameter setting, repeat the process a few times, and average the result at this parameter value. Since each repetition requires running the learning algorithm on the whole dataset, we randomly select 10% of the system's users for parameter tuning purposes in order to speed up this tuning process.

Figure 5.4a shows the ROC curve at different values of λ. When $\lambda = 0$, the least AUC was obtained because of the overfitting problem. We set λ to 0.2 since



(a) (b)

Figure 5.4: (a) Impact of varying the regularization constant λ on AUC. (b) Impact of varying both the social θ and topical β coefficients on AUC.

the AUC started to get higher values as λ increases until $\lambda = 0.2$ and dropped down thereafter. Regarding θ and β, we found, by inspection, that the system's performance degrades considerably when both parameters have values above 1. Therefore, we varied their values from 0 to 1 and jointly studied their impact on the AUC. The highest AUC=0.768 is obtained when $\theta = 0.8$ and $\beta = 0.4$ as illustrated in Figure 5.4b. Figure 5.5 gives the accuracy at the hold-out dataset with varying σ from 0 to 150.

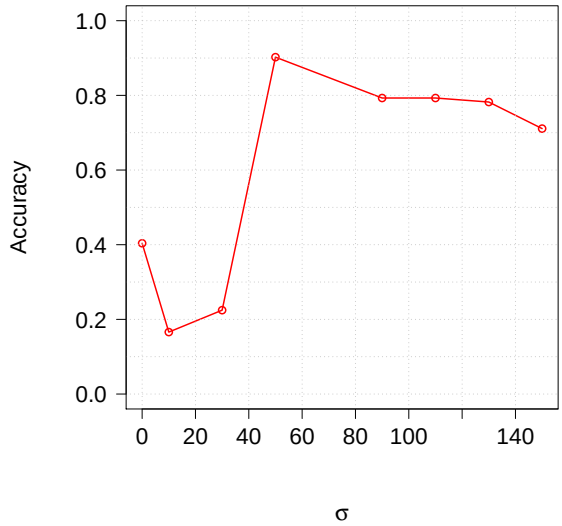


Figure 5.5: Impact of varying the standard deviation σ on accuracy

The highest value of 0.902 is achieved when $\sigma = 50$. The learning process continues until the value of AUC remains stable in consecutive iterations of the GD algorithm.

5.6.2 Experimental Results

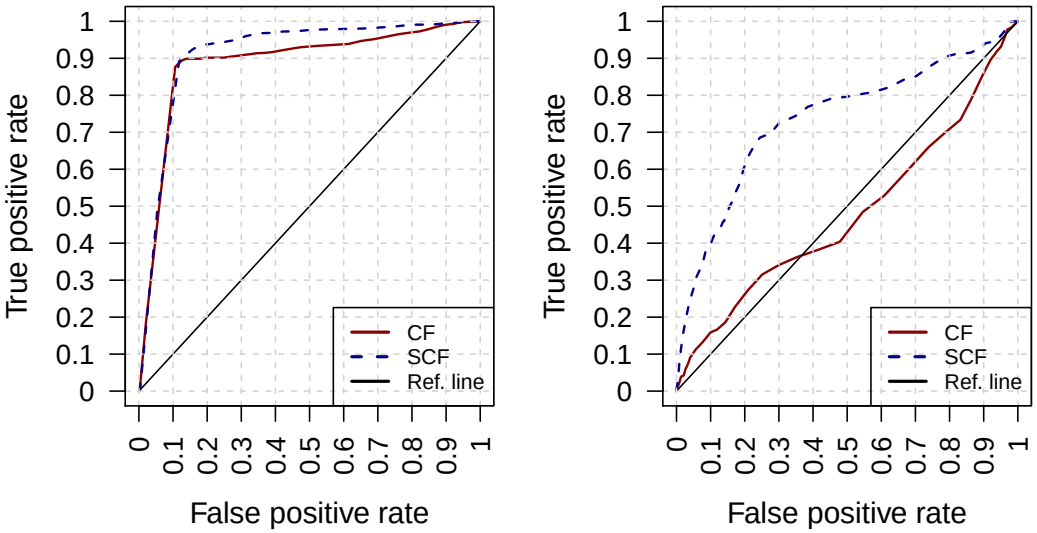
Since our proposed event recommender system is a combination of the latent factor model and three contextual features, we concentrate on showing that our hybrid model is suitable for the problem of recommending future events and it outperforms any single method. Moreover, we aim at verifying the effectiveness of each single feature. With these goals in mind, the adopted baselines in this work are listed as follows:

- **Collaborative Filtering (CF)**: It is the basic model-based CF that is built based on the regularized SVD discussed in Section 5.3.2, which is commonly used to extract latent factors. No contextual feature is used in CF.
- **Social-aware Collaborative Filtering (SCF)**: This method only considers social features as an enrichment to the basic CF. We will demonstrate the impact of this method on handling the cold-start problem caused by new users having no attendance history.
- **Topical-aware Collaborative Filtering (TCF)**: Likewise, this method only accounts for the topical feature and ignores the rest. As we will show later in this section, this method helps uncover the semantic relationship between events and thus leads to a better recommendation for future events.
- **Spatio-temporal-aware Collaborative Filtering (STCF)**: STCF only considers the spatio-temporal features and ignores others.
- **Context-aware Collaborative Filtering (CCF)**: This methods represents our hybrid method that integrates our proposed features into the regularized SVD-based model as described in Section 5.4.

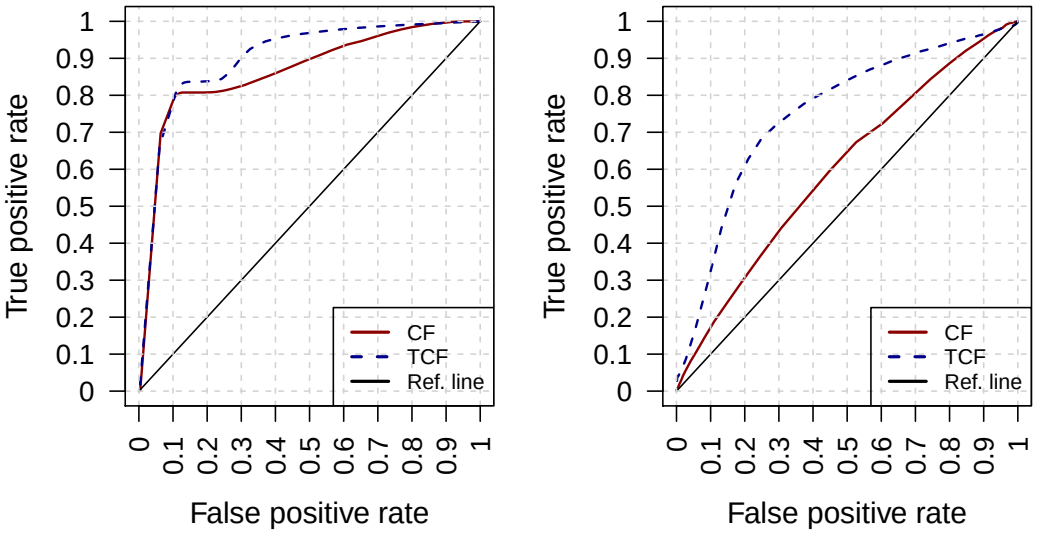
AUC and Accuracy

Figure 5.6a shows the ROC curve depicting the advantage of SCF over CF using NUd (left) and NUh (right) datasets. SCF scores better that CF in both cases, indicating that the social feature has a key role in mitigating the cold-start problem, in particular, when new users having no event attendance history are encountered. On the other hand, the topical feature scores best in terms of AUC in the NE scenario, i.e., when future events are sparsely (or even not) assigned to be attended by users. As can be seen in Figure 5.6b, TCF has led to an increase in AUC by 0.035 for NEd and 0.153 for NEh. This is because the topical feature uncovers the semantic relationship

between past and future events, and thus, the recommended future events will be semantically-consistent with the event that users attended in the past.



(a) left: ROC curve from NUd dataset, right: ROC curve from NUh dataset



(b) left: ROC curve from NEd dataset, right: ROC curve from NEh dataset

Figure 5.6: ROC curves showing the impact of individual features using different scenarios. (a) The impact of the social feature using the NU scenario. (b) The impact of the topical feature using the NE scenario. A classification model is better if its ROC curve is closer to the upper-left corner. The ROC curve of the random classifier is represented by the reference line in the diagonal.

Figure 5.7 illustrates the role our model CCF plays in improving the performance of the recommendation system over CF when the NUE scenario is considered. This scenario dictates the existence of new users and new (future) events in the dataset,

which is a real-world scenario. As can be inferred from the figure, CCF outperforms CF in case of both NUEd and NUEh. This means that CCF has a good classification ability when applied to the real and extreme scenarios where the attendance matrix have entries for new users and upcoming events.

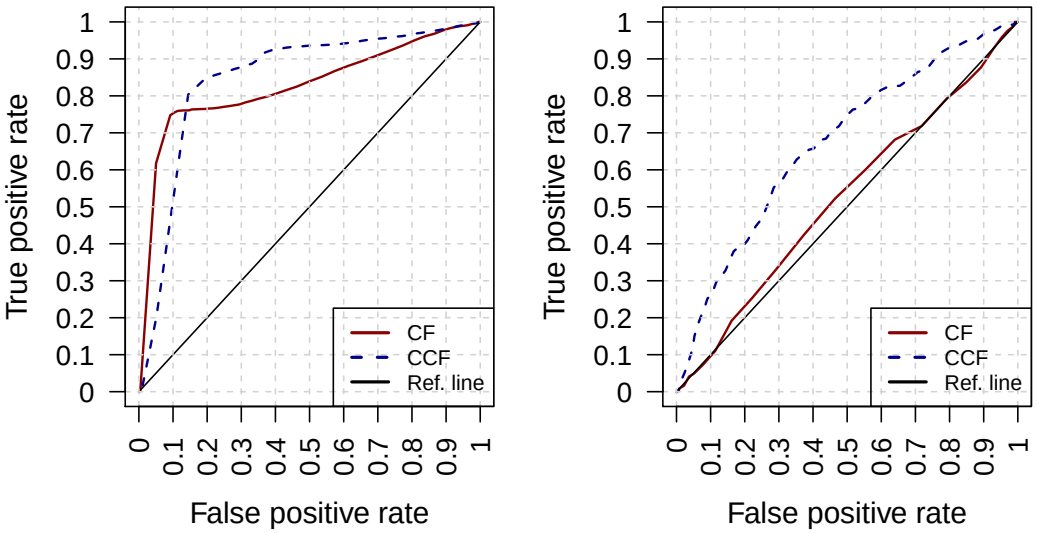


Figure 5.7: ROC curves showing the impact of CCF with respect to the NUE scenario. ROC curve from NUEd (Left) and ROC curve from NUEd (right).

Table 5.3 lists the AUC values and the accuracy of the five alternative methods under all scenarios: NU, NE, and NUE. As for NU, SCF scores best for both AUC and accuracy which proves the effectiveness of the social feature. This is because CCF consider other contextual aspects. Regarding the NE scenario, TCF shows superiority since it tries to minimize the distance between the factor vectors of past and future events. We notice that our CCF did not score best under both NU and NE, but came in the second place. This can be justified by the fact that CCF accounts for multiple contextual features; however, each of NU and NE scenarios entails reseting some of the entries of either users or events, which can be easily recovered by a certain contextual feature. For example, the entries of some users are hidden under the NU scenario. The social feature could easily capture the relationship between users and enables the suggestion of future events to new users whose group members have previously attended. It is noteworthy that event recommendation is not a simple retrieval problem. Therefore, we should go beyond recovering the hidden entries and investigate the top-k recommended events as discussed later in this section.

Finally, our hybrid model CCF outperforms the other methods under the real-world scenario NUE with AUC=0.671 and accuracy=0.879, where many new users and future events exist. This implies that the entire set of contextual features can

Table 5.3: Comparing various methods built with all features and those built with no or individual features using different scenarios. The reported results are retrieved based on the hold-out datasets of each scenario, i.e., NUh, NEh, and NUEh.

		CF	SCF	TCF	STCF	CCF
NU	AUC	0.478	**0.7368**	0.548	0.503	0.723
	Accuracy	0.863	**0.911**	0.902	0.892	0.882
NE	AUC	0.59	0.528	**0.7519**	0.594	0.722
	Accuracy	0.679	0.383	**0.814**	0.738	0.791
NUE	AUC	0.51	0.506	0.517	0.567	**0.671**
	Accuracy	0.465	0.776	0.843	0.715	**0.879**

collaboratively improve the performance of the classifier. STCF comes next, showing its effectiveness in such a strict scenario. Moreover, the traditional CF did not perform well under all scenarios, which highlights the important role the contextual features play in adapting CF for the unique nature of event recommendation.

Precision at top-k events

We turn here to evaluate the top-k recommended events and to investigate whether they math the preferences of corresponding users using *prec*@k measure.

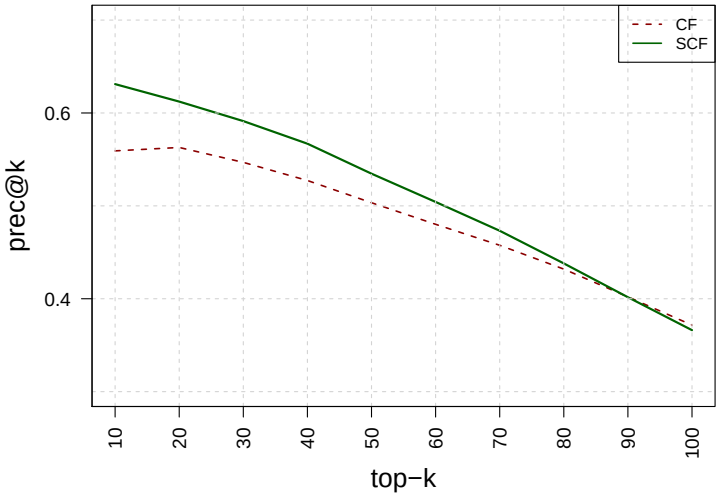


Figure 5.8: Comparison between CF and SCF in terms of *prec*@k after running the system on the NUEd dataset.

Figure 5.8 illustrates the impact of the social feature on *prec@k* after running the system on NUEd dataset. It is obvious that the social feature leads to an increase in the number of events that are ranked top and are attended by users from the same community (group). In Figure 5.9, we plot the values of *prec@k* by varying k from

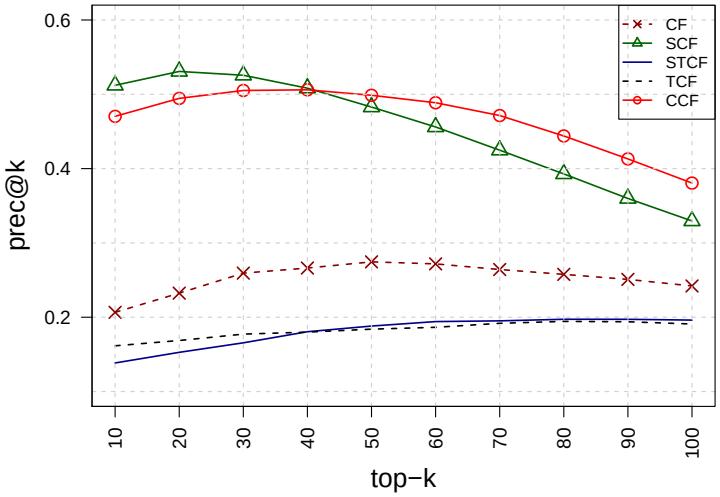


Figure 5.9: Comparison between the different alternative methods in terms of *prec@k* using dataset NUEh

10 to 100. The results are collected after running the system under the NUE scenario on the NUEh dataset. As can be seen, both SCF and CCF have in general close values of *prec@k*, which are higher than what achieved by other methods. However, SCF scores better than CCF for $k = 10, 20$ and 30. This is because SCF only focuses on the social features and ignores the impact of others as described in Section 5.6.1. The methods of STCF and TCF, relying on the spatio-temporal and topical features, respectively, perform worse than the traditional CF since both of them direct the learning process to cover individual aspects that do not serve the social-oriented *prec@k* measure. For instance, STCF favors spatially-close events that might not be attended by users belonging to the same group. As a consequence, CCF can be considered as a compromise event recommender systems that jointly serves different aspects of user preferences.

5.6.3 Evaluation of Runtime

In this section, we report the the runtime of the parameter estimation step for a number methods, namely, CCF, CF, SCF, and TCF. The method STCF is excluded from this evaluation because it does not require summing over users or events, and hence, will act in a way similar to CF. The experimental evaluation is conducted

using a platform with Intel(R) Xeon(R) CPU (2.27 GHz), 16 cores, and 64 GB main memory. We ran the experiments under Ubuntu 10.04.4 LTS (with Java 6 framework). As can be seen in Figure 5.10, The runtime required for each iteration of each method

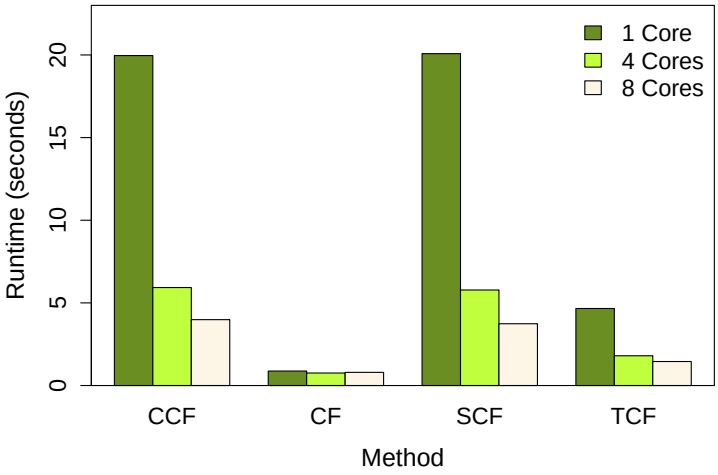


Figure 5.10: Average runtime required by each method for one iteration of the learning process.

is averaged and reported for three scenarios, i.e., using 1 core, 4 cores, and 8 cores of our multi-core platform. The reduction in runtime when using 4 cores instead of 1 core is as expected. However, Using 8 cores is not reducing the required runtime at the same rate. This can be justified by the existence of shared data that is concurrently accessed by multiple cores, a phenomenon that imposes conducting synchronization mechanisms that cause a delay in execution. Interestingly, the use of a multi-core computation does not serve CF because it is less complex than others and does not have regularization terms that involve iterating over users and/or events to update the model's parameters. One important aspect that can greatly ensure a scalable recommendation system is the use of *stochastic gradient descent*, a concept we leave for future work.

5.7 Summary

In this chapter, we addressed the event recommendation problem. Given a certain user, the proposed model generates a ranked list of future events that suit his/her preferences. For this type of recommendations, we chose to employ model-based collaborative filtering, as a well-established recommendation paradigm. Our model can thus uncover latent features characterizing both users and events, which is an

important aspect that is not possible with using only content filtering. However, collaborative filtering suffers from the cold-start problem, in particular in the context of event recommendation. Unfortunately, a small number of events are usually ranked or assigned as "attended", which exaggerates the cold-start problem. On the other hand, events are time-specific in the sense that only future events can be suggested to users. Moreover, the spatial and temporal dimensions of events can not be ignored during recommendation, which are not explicitly modeled by the traditional collaborative filtering. These challenges for event recommendation require a careful adaptation of collaborative filtering models for better recommendation results.

To this end, we proposed a context-aware event recommender system that accounts for social, topical, and spatio-temporal features. The well-known matrix factorization is employed to realize our model, where the regularized Singular Value Decomposition (SVD) is used to estimate the model's parameters. The contextual features we discussed above can be viewed as regularization terms for this regularized SVD. These regularization terms, in turn, can direct the learning process to produce recommendations that better match the preferences of users. The experimental evaluation of our proposed model, which is conducted using a Meetup dataset, revealed the important role the proposed contextual features play in providing personalized event recommendations and in handling the cold-start problem.

Chapter 6

Conclusions and Future Work

Detecting real-world events that unfold over space and time from streaming microblogs provides actionable and situational knowledge to a variety of applications, e.g., event recommendation and context-aware information retrieval. To best utilize these applications, event information should be extracted and incorporated in a timely manner. Hence, event detection from social media must efficiently and accurately uncover event-related information that is buried within large amounts of noisy and mundane content, e.g., meaningless, polluted, and rumor messages. As a consequence, this has motivated extensive research in various fields such as databases, data mining, and natural language processing to focus on identifying and tracking events from social media.

In this thesis, we gave particular attention to localized events, i.e., real-world events occurring at certain geographic places. The fundamental tasks defined and studied are (1) localized event detection and tracking and (2) localized event recommendation. In the following, we first highlight our key results of this thesis and then give an outlook for further studies.

6.1 Summary

Current solutions to the problem of event detection in social media suffer from many limitations. In general, these solutions do not achieve a good trade-off between accuracy and scalability, are not robust against the high-levels of noise and spatio-temporal problems of keywords, and/or do not consider all dimensions of events, i.e., semantic, location, and time. To overcome this, we proposed a novel 4-stage framework called LocEvent that is the first to tackle all these aspects at once. LocEvent is built upon the sliding window model and thus incrementally processes a stream of incoming

messages and generates event clusters that are updated (tracked) as time progresses. It also employs a multi-stage noise elimination mechanism that starts, in Stage-1, by extracting event-related keywords, followed by excluding non-local keywords in Stage-2. Furthermore, in Stage-3, it adopts a noise-resilient clustering algorithm, where, in Stage-4, each cluster is scored so that non-event clusters get low scores. In addition, LocEvent estimates, for each detected event, a set of informative keywords, a time interval, and a location at a fine-grained resolution.

For keyword extraction, Stage-1 of LocEvent and a fundamental task in feature-pivot detection, we proposed a novel framework KeyPicker that exploits a two-level temporal index to organize messages based on the attached timestamps and the contained words. KeyPicker tackles two temporal problems, namely, temporal heterogeneity and temporal outliers. To handle heterogeneity, we utilize the concept of time-aware baseline and applied the discrepancy paradigm for an efficient and reliable keyword extraction. Keywords are identified by quantifying and integrating two word characteristics: *burstiness* and *recurrence*, which helps mitigate the consequences of temporal outliers. Finally, keywords are assigned dynamic scores that evolve over time, capturing the life cycle of each keyword.

Since local keywords are the main building blocks of localized events, we employed, in Stage-2 of LocEvent, an entropy-based method to distinguish local keywords among others having broad geographic extents. However, the direct application of this method suffers from two major problems: spatial outliers and spatial sparsity, which we handle by two novel regularization procedures and a non-parametric kernel density estimation, respectively. To ensure scalability given a huge stream of incoming messages, we applied (1) a stage-wise pruning of words, (2) a hierarchical space-partitioning index structure, and (3) an incremental clustering algorithm for a single-pass processing of event entities. To evaluate LocEvent, different datasets collected from Twitter were used, and the analysis of the experimental results shows that our frameworks outperform existing methods in terms of the relevancy and precision of the mined event patterns.

Finally, as an application, we proposed a context-aware event recommender system on the basis of model-based collaborative filtering that is realized using the well-known matrix factorization. We utilized three contextual features: social, topical, and spatio-temporal features, which are leveraged to adapt the regularized Singular Value Decomposition (SVD) model. A dataset collected from Meetup was used to evaluate the model and the obtained results showed that the system provides recommendations

that better suit the interests of users and are effective in the presence of the cold-start problem.

6.2 Future Work

In the following, we outline a number of promising ideas emerged during the study, which could be considered to extend our work:

- **Geo-tagging of microblogs.** Only a small percentage of published microblogs are geo-tagged with geo-coordinates. This exaggerates the problem of spatial sparsity and lowers the chance of detecting small-scale events. Therefore, estimating location information for non-geo-tagged microblogs can improve the performance of LocEvent and helps in detecting more events. Recently, several research studies have considered the problem of microblog geo-tagging, leveraging several aspects including social relationships [22], the content of microblogs [49], or embedded geo-references [154]. Recall that, in Chapter 4, geo-references (place names) are extracted from goe-tagged microblogs and correlated with event-related keywords in order to generate geo-filters. In fact, observing a geo-reference along with at least one of its correlated keywords in a non-geo-tagged microblog can accurately hint to the microblog's location. Based on this, LocEvent can be extended by including a geo-tagger module before the stage of local keyword identification. Other features such as social links and address information of users can be integrated within a statistical model, e.g., the maximum entropy model, for better geo-tagging.

- **Extracting temporal expressions.** LocEvent depends on both the creation time of generated clusters and the temporal characteristics of keywords to estimate the time intervals of detected events. Extracting temporal expressions from the content of microblogs and identifying co-occurrence patterns between these expressions and event-related keywords provide a better estimation of the time interval of a detected event. Existing systems for temporal tagging, e.g., HeidelTime [146] can be used in the keyword extraction stage of KeyPicker to enrich the temporal index with temporal expressions. These expressions, when assigned to event clusters, can be utilized to adjust the estimated time intervals. Applying a temporal tagger affects the scalability of LocEvent. That is, we empirically found that utilizing temporal taggers is infeasible without im-

proving the computational efficiency of the system using strategies like parallel processing.

- **Parallelized LocEvent.** As illustrated in Chapter 4, words retrieved from the temporal index are processed separately in the first two stages of LocEvent. Therefore, these words can be distributed among multiple cores (threads). Each of these processing threads extracts keywords, identifies local ones, and then appends these local keywords to a synchronized queue for further processing, i.e., clustering and scoring.

- **Classifying event clusters.** In the clustering stage of LocEvent, clusters are formed by grouping spatially-close local keywords. In spite of using both a multi-stage noise elimination approach and a noise-resilient clustering mechanism, there are still some clusters that do not correspond to actual real-world events. A supervised classification method can be used to filter out such clusters. For this, features describing event clusters are to be extracted and annotated to prepare a training dataset. Then, a classifier is trained to distinguish event from non-event clusters. As discussed in Chapter 4, the score estimated for each cluster is considered an important feature for this classification task as non-event clusters tend to obtain small values compared to event clusters.

Bibliography

[1] PearAnalytics. Twitter study, August 2009. `http://pearanalytics.com/wp-content/uploads/2012/12/Twitter-Study-August-2009.pdf`. Accessed Jan. 2015.

[2] Hamed Abdelhaq and Michael Gertz. On the locality of keywords in twitter streams. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoStreaming*, IWGS '14, pages 12–20, 2014.

[3] Hamed Abdelhaq, Michael Gertz, and Christian Sengstock. Spatio-temporal characteristics of bursty words in twitter streams. In *Proceedings of the 21th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '13, pages 149–158, 2013.

[4] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. Eventweet: Online localized event detection from twitter. *Proceedings of the VLDB Endowment*, 6(12):1326–1329, 2013.

[5] Dhekar Abhik and Durga Toshniwal. Sub-event detection during natural hazards using features of social media data. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 783–788, 2013.

[6] Charu C. Aggarwal, editor. *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*. Springer, 2007.

[7] Charu C Aggarwal. An introduction to data streams. In *Data Streams*, pages 1–8. Springer, 2007.

[8] Charu C. Aggarwal and Karthik Subbian. Event detection in social streams. In *SDM*, pages 624–635, 2012.

[9] Charu C. Aggarwal and ChengXiang Zhai. *A Survey of Text Clustering Algorithms*. Springer US, 2012.

[10] M-Dyaa Albakour, Craig Macdonald, and Iadh Ounis. Identifying local events by using microblogs as social sensors. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, OAIR '13, pages 173–180, 2013.

[11] J Allan. *Topic detection and tracking: event-based information organization*, volume 12. Springer, 2002.

[12] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, Yiming Yang, Umass Amherst, and Dragon Systems. Topic detection and tracking pilot study final report. 1998.

[13] James Allan, Victor Lavrenko, and Hubert Jin. First story detection in TDT is hard. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, pages 374–381, 2000.

[14] James Allan, Ron Papka, and Victor Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 37–45, 1998.

[15] Foteini Alvanaki, Sebastian Michel, Krithi Ramamritham, and Gerhard Weikum. See what's enblogue: Real-time emergent topic identification in social media. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 336–347, 2012.

[16] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R Motwani, U. Srivastava, and J. Widom. Stream: The stanford data stream management system. Technical Report 2004-20, Stanford InfoLab, 2004.

[17] Akinori Asahara, Kishiko Maruyama, Akiko Sato, and Kouichi Seto. Pedestrian-movement prediction based on mixed markov-chain model. In *Proceedings of the 19th ACM International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '11, pages 25–33, 2011.

[18] Farzindar Atefeh and Wael Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 2013.

[19] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, 2002.

[20] Brian Babcock, Mayur Datar, and Rajeev Motwani. Sampling from a moving window over streaming data. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 633–634, 2002.

[21] Lars Backstrom, Jon Kleinberg, Ravi Kumar, and Jasmine Novak. Spatial variation in search engine queries. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 357–366, 2008.

[22] Lars Backstrom, Eric Sun, and Cameron Marlow. Find me if you can: Improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 61–70, 2010.

[23] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., 1999.

[24] Jie Bao, Yu Zheng, and Mohamed F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 199–208, 2012.

[25] Hila Becker. *Identification and characterization of events in social media*. PhD thesis, Columbia University, 2011.

[26] Hila Becker, Dan Iter, Mor Naaman, and Luis Gravano. Identifying content for planned events across social media sites. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 533–542, 2012.

[27] Hila Becker, Mor Naaman, and Luis Gravano. Learning similarity metrics for event identification in social media. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 291–300, 2010.

[28] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. In *ICWSM*, 2011.

[29] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM International Conference on Management of Data*, SIGMOD '90, pages 322–331, 1990.

[30] Edward Benson, Aria Haghighi, and Regina Barzilay. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT '11, pages 389–398, 2011.

[31] P.J. Bickel and K.A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*. Number v. 1 in Holden-Day series in probability and statistics. Prentice Hall, 2001.

[32] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[33] David M. Blei and John D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 113–120, 2006.

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

[35] Alexander Boettcher and Dongman Lee. Eventradar: A real-time local event detection scheme using twitter stream. In *IEEE International Conference on Green Computing and Communications (GreenCom)*, pages 358–367, 2012.

[36] Ioannis Boutsis, Stavroula Karanikolaou, and Vana and Kalogeraki. Personalized event recommendations using social networks. In *the 16th IEEE International Conference on Mobile Data Management*, MDM '15, pages 43–48, 2015.

[37] Thorsten Brants, Francine Chen, and Ayman Farahat. A system for new event detection. In *Proceedings of the 26th Annual International ACM Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 330–337, 2003.

[38] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[39] Ceren Budak, Theodore Georgiou, Divyakant Agrawal, and Amr El Abbadi. Geoscope: Online detection of geo-correlated information trends in social networks. *Proceedings of the VLDB Endowment*, 7(4), 2013.

[40] Khoo Khyou Bun and M. Ishizuka. Topic extraction from news archive using tf*pdf algorithm. In *Proceedings of the Third International Conference on Web Information Systems Engineering, WISE '02*, pages 73–82, 2002.

[41] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, pages 4:1–4:10, 2010.

[42] Deepayan Chakrabarti and Kunal Punera. Event Summarization Using Tweets. *ICWSM*, pages 66–73, 2011.

[43] Chazelle. *The discrepancy method : randomness and complexity*. Cambridge University Press, 2000.

[44] Chien Chin Chen, Yao-Tsung Chen, Yeali Sun, and Meng Chang Chen. Life cycle modeling of news events using aging theory. In *Machine Learning: ECML 2003*, pages 47–59. Springer, 2003.

[45] Kuan-Yu Chen, Luesak Luesukprasert, and Seng-cho T. Chou. Hot topic extraction based on timeline analysis and multidimensional sentence modeling. *IEEE Trans. on Knowl. and Data Eng.*, 19(8):1016–1025, August 2007.

[46] L. Chen and A. Roy. Event detection from flickr data through wavelet-based spatial analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 523–532, 2009.

[47] Wei Chen, Chun Chen, Li-jun Zhang, Can Wang, and Jia-jun Bu. Online detection of bursty events and their evolution in news streams. *Journal of Zhejiang University SCIENCE C*, 11(5):340–355, 2010.

[48] Ying Chen, Bo Xu, Hongwei Hao, Shiyu Zhou, and Jie Cao. User-defined hot topic detection in microblogging. In *Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service*, ICIMCS '13, pages 183–186, 2013.

[49] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. You are where you tweet: A content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 759–768, 2010.

[50] Rumi Chunara, Jason R Andrews, and John S Brownstein. Social and news media enable estimation of epidemiological patterns early in the 2010 haitian cholera outbreak. *The American Journal of Tropical Medicine and Hygiene*, 86(1):39–45, 2012.

[51] Christopher Cieri, Stephanie Strassel, David Graff, Nii Martey, Kara Rennert, and Mark Liberman. Corpora for topic detection and tracking. In James Allan, editor, *Topic Detection and Tracking*, volume 12 of *The Information Retrieval Series*, pages 33–66. Springer US, 2002.

[52] Chris Cornelis, Xuetao Guo, Jie Lu, and Guanquang Zhang. A fuzzy relational approach to event recommendation. In *IICAI*, volume 5, pages 2231–2242, 2005.

[53] Anish Das Sarma, Alpa Jain, and Cong Yu. Dynamic relationship and event discovery. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 207–216, 2011.

[54] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows: (extended abstract). In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 635–644, 2002.

[55] Cedric De Boom, Steven Van Canneyt, and Bart Dhoedt. Semantics-driven event clustering in twitter feeds. In Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie, editors, *Proceedings of the 5th Workshop on Making Sense of Microposts*, volume 1395, pages 2–9, 2015.

[56] Bertrand De Longueville, Robin S. Smith, and Gianluca Luraschi. "OMG, from here, i can see the flames!": A use case of mining location based social networks to acquire spatio-temporal data on forest fires. In *Proceedings of the 2009 International Workshop on Location Based Social Networks*, LBSN '09, pages 73–80, 2009.

[57] David P. Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *Journal of Computer and System Sciences*, 52(3):453 – 470, 1996.

[58] Ahmed Elbery, Mustafa ElNainay, Feng Chen, Chang-Tien Lu, and Jeffrey Kendall. A carpooling recommendation system based on social vanet and geo-social data. In *Proceedings of the 21st ACM International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '13, pages 556–559, 2013.

[59] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB*, volume 98, pages 323–333, 1998.

[60] Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. Automatic creation of domain templates. In *Proceedings of the COLING/ACL 2006*, pages 207–214, 2006.

[61] Jonathan G Fiscus and George R Doddington. Topic detection and tracking evaluation overview. *Topic detection and tracking*, pages 17–31, 2002.

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

[63] S. Funk. Netflix update: Try this at home, December 2006. `http://sifter.org/~simon/journal/20061211.html`. Accessed June 2015.

[64] Wei Gao and Guohong Cao. Fine-grained mobility characterization: steady and transient state behaviors. In *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*, pages 61–70, 2010.

[65] Xingyu Gao, Juan Cao, Qin He, and Jintao Li. A novel method for geographical social event detection in social media. In *Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service*, pages 305–308, 2013.

[66] Johannes Gehrke, Flip Korn, and Divesh Srivastava. On computing correlated aggregates over continual data streams. In *Proceedings of the 2001 ACM International Conference on Management of Data*, SIGMOD '01, pages 13–24, 2001.

[67] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[68] Genevieve Gorrell and Brandyn Webb. Generalized hebbian algorithm for incremental latent semantic analysis. In *INTERSPEECH*, pages 1325–1328, 2005.

[69] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[70] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[71] Adrien Guille and Cécile Favre. Event detection, tracking, and visualization in twitter: a mention-anomaly-based approach. *Social Network Analysis and Mining*, 5(1), 2015.

[72] Jheser Guzman and Barbara Poblete. On-line relevant anomaly detection in the twitter stream: An efficient bursty keyword detection model. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, ODD '13, pages 31–39, 2013.

[73] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[74] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[75] Dan He and D. Stott Parker. Topic dynamics: an alternative model of bursts in streams of topics. In *Proceedings of the 16th ACM international conference on Knowledge discovery and data mining*, KDD '10, pages 443–452, 2010.

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

[77] Qi He, Kuiyu Chang, Ee-Peng Lim, and Jun Zhang. Bursty feature representation for clustering text streams. In *SDM*, pages 491–496, 2007.

[78] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, 1999.

[79] Courtenay Honey and Susan C Herring. Beyond microblogging: Conversation and collaboration via twitter. In *42nd Hawaii International Conference on System Sciences.*, HICSS'09, pages 1–10, 2009.

[80] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. Discovering geographical topics in the twitter stream. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 769–778, 2012.

[81] Liangjie Hong, Byron Dom, Siva Gurumurthy, and Kostas Tsioutsiouliklis. A time-dependent topic model for multiple text streams. In *Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 832–840, 2011.

[82] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, pages 49–56, 2008.

[83] Jin Huang and Charles X Ling. Using auc and accuracy in evaluating learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 17(3):299–310, 2005.

[84] R.A. Jarvis and Edward A. Patrick. Clustering using a similarity measure based on shared near neighbors. *Computers, IEEE Transactions on*, C-22(11):1025–1034, Nov 1973.

[85] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: Understanding microblogging usage and communities. In *Proceedings of the 9th Workshop on Web Mining and Social Network Analysis*, WebKDD/SNA-KDD '07, pages 56–65, 2007.

[86] Xin Jin, Andrew Gallagher, Liangliang Cao, Jiebo Luo, and Jiawei Han. The wisdom of social multimedia: Using flickr for prediction and forecast. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 1235–1244, 2010.

[87] Krishna Y. Kamath, James Caverlee, Kyumin Lee, and Zhiyuan Cheng. Spatio-temporal dynamics of online memes: A study of geo-tagged tweets. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 667–678, 2013.

[88] Andreas M. Kaplan and Michael Haenlein. Users of the world, unite! the challenges and opportunities of social media. *Business Horizons*, 53(1):59–68, 2010.

[89] M. Kayaalp, T. Ozyer, and S.T. Ozyer. A collaborative and content based event recommendation system integrated with data collection scrapers and services at a social networking site. In *International Conference on Advances in Social Network Analysis and Mining, ASONAM '09.*, pages 113–118, 2009.

[90] Jon Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the Eighth ACM International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 91–101, 2002.

[91] April Kontostathis, LeonM. Galitsky, WilliamM. Pottenger, Soma Roy, and DanielJ. Phelps. A survey of emerging trend detection in textual data mining. In MichaelW. Berry, editor, *Survey of Text Mining*, pages 185–224. Springer New York, 2004.

[92] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[93] Giridhar Kumaran and James Allan. Text classification and named entities for new event detection. In *Proceedings of the 27th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 297–304, 2004.

[94] Elizabeth Kwan, Pei-Ling Hsu, Jheng-He Liang, and Yi-Shin Chen. Event identification for social streams using keyword-based evolving graph sequences. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 450–457, 2013.

[95] Theodoros Lappas, Benjamin Arai, Manolis Platakis, Dimitrios Kotsakos, and Dimitrios Gunopulos. On burstiness-aware search for document sequences. In *KDD*, pages 477–486, 2009.

[96] Theodoros Lappas, Marcos R Vieira, Dimitrios Gunopulos, and Vassilis J Tsotras. On the spatiotemporal burstiness of terms. *Proceedings of the VLDB Endowment*, 5(9):836–847, 2012.

[97] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the Fifth ACM International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 16–22, 1999.

[98] Chung-Hong Lee. Mining spatio-temporal information on microblogging streams using a density-based online clustering method. *Expert Systems with Applications*, 39(10):9623 – 9641, 2012.

[99] Chung-Hong Lee, Chih-Hong Wu, and Tzan-Feng Chien. Burst: a dynamic term weighting scheme for mining microblogging messages. In *Proceedings of the 8th international conference on Advances in neural networks*, pages 548–557, 2011.

[100] Ryong Lee and Kazutoshi Sumiya. Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, LBSN '10, pages 1–10, 2010.

[101] Sungjun Lee, Sangjin Lee, Kwanho Kim, and Jonghun Park. Bursty event detection from text streams for disaster management. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 679–682, 2012.

[102] Justin J Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F Mokbel. Lars: A location-aware recommender system. In *28th International Conference on Data Engineering*, ICDE '12, pages 450–461, 2012.

[103] Chenliang Li, Aixin Sun, and Anwitaman Datta. Twevent: Segment-based event detection from tweets. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 155–164, 2012.

[104] Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. Twiner: Named entity recognition in targeted twitter stream. In *Proceedings of the 35th International Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 721–730, 2012.

[105] Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang. Tedas: A twitter-based event detection and analysis system. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 1273–1276, 2012.

[106] Zhiwei Li, Bin Wang, Mingjing Li, and Wei-Ying Ma. A probabilistic model for retrospective news event detection. In *Proceedings of the 28th Annual International Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 106–113, 2005.

[107] Xingjie Liu, Qi He, Yuanyuan Tian, Wang-Chien Lee, John McPherson, and Jiawei Han. Event-based social networks: Linking the online and offline social worlds. In *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1032–1040, 2012.

[108] Rui Long, Haofen Wang, Yuqiang Chen, Ou Jin, and Yong Yu. Towards effective event detection, tracking and summarization on microblog data. In *Web-Age Information Management*, pages 652–663. Springer, 2011.

[109] Amr Magdy, Louai Alarabi, Saif Al-Harthi, Mashaal Musleh, Thanaa M. Ghanem, Sohaib Ghani, and Mohamed F. Mokbel. Taghreed: A system for querying, analyzing, and visualizing geotagged microblogs. In *Proceedings of the 22nd ACM International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 163–172, 2014.

[110] Amr Magdy, Mohamed F. Mokbel, Sameh Elnikety, Suman Nath, and Yuxiong He. Mercury: A memory-constrained spatio-temporal real-time search on microblogs. In *Proceedings of the 2014 IEEE 30th International Conference on*, ICDE '14, pages 172–183, 2014.

[111] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[112] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 227–236, 2011.

[113] Michael Mathioudakis, Nilesh Bansal, and Nick Koudas. Identifying, attributing and describing spatial bursts. *Proceedings of the VLDB Endowment*, 3(1-2):1091–1102, 2010.

[114] Michael Mathioudakis and Nick Koudas. Twittermonitor: Trend detection over the twitter stream. In *Proceedings of the ACM International Conference on Management of Data*, SIGMOD '10, pages 1155–1158, 2010.

[115] P. McFedries. Technically speaking: All a-twitter. *IEEE Spectr.*, 44(10):84–84, October 2007.

[116] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Database Theory-ICDT 2005*, pages 398–412. Springer, 2005.

[117] Einat Minkov, Ben Charrow, Jonathan Ledlie, Seth Teller, and Tommi Jaakkola. Collaborative future event recommendation. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 819–828, 2010.

[118] Mor Naaman, Jeffrey Boase, and Chih-Hui Lai. Is it really about me?: Message content in social awareness streams. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10, pages 189–192, 2010.

[119] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[120] Duc T. Nguyen and Jason J. Jung. Real-time event detection on social data stream. *Mobile Networks and Applications*, pages 1–12, 2014.

[121] Ruchi Parikh and Kamalakar Karlapalem. Et: Events from tweets. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 613–620, 2013.

[122] Jon Parker, Yifang Wei, Andrew Yates, Ophir Frieder, and Nazli Goharian. A framework for detecting public health trends with twitter. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 556–563, 2013.

[123] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[124] Saša Petrović, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 181–189, 2010.

[125] Yanxia Qin, Yue Zhang, Min Zhang, and Dequan Zheng. Feature-rich segment-based news event detection on twitter. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 302–310, 2013.

[126] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.

[127] Tye Rattenbury, Nathaniel Good, and Mor Naaman. Towards automatic extraction of event and place semantics from flickr tags. In *Proceedings of the 30th annual international ACM conference on Research and development in information retrieval*, SIGIR '07, pages 103–110, 2007.

[128] Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1104–1112, 2012.

[129] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill Higher Education, 5th edition, 2002.

[130] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World Wide Web*, WWW '10, pages 851–860, 2010.

[131] Gerard Salton, editor. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, 1971.

[132] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing and management*, 24(5):513–523, 1988.

[133] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[134] Hanan Samet. Applications of spatial data structures. Addison-Wesley, 1990.

[135] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th ACM International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '09, pages 42–51, 2009.

[136] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event detection and tracking in social streams. In *Proceedings of the International Conference on Weblogs and Social Media*, ICWSM '09, 2009.

[137] Christian Sengstock, Michael Gertz, Hamed Abdelhaq, and Florian Flatow. Reliable spatio-temporal signal extraction and exploration from human activity records. In *Advances in Spatial and Temporal Databases*, SSTD '13, pages 484–489. 2013.

[138] Christian Sengstock, Michael Gertz, Florian Flatow, and Hamed Abdelhaq. A probablistic model for spatio-temporal signal extraction from social media. In *Proceedings of the 21st ACM International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 274–283, 2013.

[139] Dongdong Shan, Wayne Xin Zhao, Rishan Chen, Baihan Shu, Ziqi Wang, Junjie Yao, Hongfei Yan, and Xiaoming Li. Eventsearch: a system for event discovery and retrieval on multi-type historical data. In *Proceedings of the 18th ACM international conference on Knowledge discovery and data mining*, KDD '12, pages 1564–1567, 2012.

[140] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[141] R. W. Sinnott. Virtues of the Haversine. *Sky and Telescope*, 68(2):158, 1984.

[142] A. Skovsgaard, D. Sidlauskas, and C.S. Jensen. Scalable top-k spatio-temporal term querying. In *30th IEEE International Conference on Data Engineering*, ICDE '14, pages 148–159, 2014.

[143] Julius O. Smith. *Mathematics of the Discrete Fourier Transform (DFT)*. W3K Publishing, 2007.

[144] Jan A. Snyman. *Practical mathematical optimization : an introduction to basic optimization theory and classical and new gradient-based algorithms*. Applied optimization. Springer, 2005.

[145] Pawel Sobkowicz, Michael Kaschesky, and Guillaume Bouchard. Opinion mining in social media: Modeling, simulating, and forecasting political opinions in the web. *Government Information Quarterly*, 29(4):470 – 479, 2012.

[146] Jannik Strötgen and Michael Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013.

[147] S. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, 4(2):104 – 119, 1975.

[148] Douglas Terry, David Goldberg, David Nichols, and Brian Oki. Continuous queries over append-only databases. In *Proceedings of the ACM International Conference on Management of Data*, SIGMOD '92, pages 321–330, 1992.

[149] Waldo R Tobler. A computer movie simulating urban growth in the detroit region. *Economic geography*, pages 234–240, 1970.

[150] G. Valkanas and D. Gunopulos. Location extraction from social networks with commodity software and online data. In *IEEE 12th International Conference on Data Mining Workshops*, ICDMW '12, pages 827–834, 2012.

[151] George Valkanas and Dimitrios Gunopulos. How the live web feels about events. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, pages 639–648, 2013.

[152] Maximilian Walther and Michael Kaisser. Geo-spatial event detection in the twitter stream. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, ECIR'13, pages 356–367, 2013.

[153] M. P. Wand. *Kernel smoothing*. Chapman and Hall, London New York, 1995.

[154] Kazufumi Watanabe, Masanao Ochi, Makoto Okabe, and Rikio Onai. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 2541–2544, 2011.

[155] Andreas Weiler, Marc H. Scholl, Franz Wanner, and Christian Rohrdantz. Event identification for local areas using social media streaming data. In *Proceedings of the ACM SIGMOD Workshop on Databases and Social Networks*, DBSocial '13, pages 1–6, 2013.

[156] Jianshu Weng and Bu-Sung Lee. Event detection in twitter. In *5th International AAAI Conference on Weblogs and Social Media, ICWSM '11*, 2011.

[157] Paul Whitney, Dave Engel, and Nick Cramer. Mining for surprise events within text streams. In *SDM*, pages 617–627. SIAM, 2009.

[158] Jun-Ming Xu, Aniruddha Bhargava, Robert Nowak, and Xiaojin Zhu. Socioscope: Spatio-temporal signal recovery from social media. In *Machine Learning and Knowledge Discovery in Databases*, pages 644–659. Springer, 2012.

[159] Liang Yang, Yuan Lin, and Hongfei Lin. Micro-blog hot event detection based on emotion distribution. *Journal of Chinese Information Processing*, 26(1):84–90, 2012.

[160] Yiming Yang, Tom Ault, Thomas Pierce, and Charles W. Lattimer. Improving text categorization methods for event tracking. In *Proceedings of the 23rd Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 65–72, 2000.

[161] Yiming Yang, Jaime G. Carbonell, Ralf D. Brown, Thomas Pierce, Brian T. Archibald, and Xin Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, July 1999.

[162] Yiming Yang, Tom Pierce, and Jaime Carbonell. A study of retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 28–36, 1998.

[163] Yiming Yang, Jian Zhang, Jaime Carbonell, and Chun Jin. Topic-conditioned novelty detection. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 688–693, 2002.

[164] Junjie Yao, Bin Cui, Yuxin Huang, and Yanhong Zhou. Bursty event detection from collaborative tags. *World Wide Web*, 15(2):171–195, 2012.

[165] Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu. *Social Media Mining*. Cambridge University Press, Cambridge, 2014.

[166] Haipeng Zhang, Mohammed Korayem, Erkang You, and David J. Crandall. Beyond co-occurrence: discovering and visualizing tag relationships from geo-spatial and temporal similarities. In *Proceedings of the 5th ACM international conference on Web search and data mining*, WSDM '12, pages 33–42, 2012.

[167] Wei Zhang, Jianyong Wang, and Wei Feng. Combining latent factor model with location features for event-based group recommendation. In *Proceedings of the 19th ACM International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 910–918, 2013.

[168] Xiaoming Zhang, Xiaoming Chen, Yan Chen, Senzhang Wang, Zhoujun Li, and Jiali Xia. Event detection and popularity prediction in microblogging. *Neurocomputing*, 149, Part C(0):1469 – 1480, 2015.

[169] Dejin Zhao and Mary Beth Rosson. How and why people twitter: The role that micro-blogging plays in informal communication at work. In *Proceedings of the ACM International Conference on Supporting Group Work*, GROUP '09, pages 243–252, 2009.

[170] Qiankun Zhao and Prasenjit Mitra. Event detection and visualization for social text streams. In *Proceedings of International Conference on Weblogs and Social Media*, ICWSM '07, 2007.

[171] Xiangmin Zhou and Lei Chen. Event detection over twitter social media streams. *The VLDB Journal*, 23(3):381–400, 2014.

[172] Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the Ninth ACM International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 336–345, 2003.