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. Le Van Quoc Anh .....

Born in: Hue, Vietnam .....

Oral examination: .....

Pattern Discovery from Event Data

Advisor: Prof. Dr. Michael Gertz

Abstract

Events are ubiquitous in real-life. With the rapid rise of the popularity of social media channels, massive amounts of event data, such as information about festivals, concerts, or meetings, are increasingly created and shared by users on the Internet. Deriving insights or knowledge from such social media data provides a semantically rich basis for many applications, for instance, social media marketing, service recommendation, sales promotion, or enrichment of existing data sources. In spite of substantial research on discovering valuable knowledge from various types of social media data such as microblog data, check-in data, or GPS trajectories, interestingly there has been only little work on mining event data for useful patterns.

In this thesis, we focus on the discovery of interesting, useful patterns from datasets of events, where information about these events is shared by and spread across social media platforms. To deal with the existence of heterogeneous event data sources, we propose a comprehensive framework to model events for pattern mining purposes, where each event is described by three components: context, time, and location. This framework allows one to easily define how events are related in terms of conceptual, temporal, and spatial (geographic) relationships. Moreover, we also take into account hierarchies for contexts, time, and locations of events, which naturally exist as useful background knowledge to derive patterns at different levels of abstraction and granularity. Based on this framework, we focus on the following problems: (i) mining interval-based event sequence patterns, (ii) mining periodic event patterns, and (iii) extracting semantic annotations for locations of events. Generally, the first two problems consider correlations of events whereas the last one takes correlations of event components into account.

In particular, the first problem is a generalization of mining sequential patterns from traditional data, where patterns representing complex temporal relationships among events can be discovered at different levels of abstraction and granularity. The second problem is to find periodic event patterns, where a notion of relaxed periodicity is formulated for events as well as for groups of events that co-occur. The third problem is to extract semantic annotations for locations on the basis of exploiting correlations of contexts, time, and locations of events. For the three problems above, we respectively propose novel and efficient approaches. Our experiments clearly indicate that extracted patterns and knowledge can be well utilized in various useful tasks, such as event prediction, semantic search for locations, or topic-based clustering of locations.

Zusammenfassung

Events sind im täglichen Leben allgegenwärtig. Mit der rasant wachsenden Beliebtheit sogenannter *Social Media* Anwendungen werden sehr große Datenmengen über Events von Benutzern im Internet generiert und geteilt. Beispiele hierfür sind Informationen über Festivals, Konzerte, oder Meetings. Aus diesen Datensätzen der Social Media Anwendungen abgeleitetes Wissen stellt eine semantisch reiche Grundlage für zahlreiche Anwendungen dar, beispielsweise für Social Media Marketing, Produkt- und Dienstleistungsempfehlungen, aber auch für das semantische Anreichern vorhandener Datenquellen. Trotz weitreichender Forschung um wertvolles Wissen aus verschiedenen Social Media Daten (z.B. Mikroblogs, Check-in-Daten, und GPS-Trajektorien) zu extrahieren, gibt es aktuell nur wenige Arbeiten, die sich mit der Extraktion von Mustern aus Eventdaten beschäftigen.

Der Schwerpunkt dieser Arbeit liegt auf dem Auffinden interessanter, nützlicher Muster aus Event-Datensätzen, wobei die Informationen über Events über unterschiedliche Social Media Anwendungen geteilt und verbreitet werden. Zur Verarbeitung der heterogenen Datenquellen stellen wir ein umfassendes Framework vor, das Events für die Mustererkennung modelliert. Ein Event besteht dabei aus drei Komponenten: Kontext-, Zeit- und Ortsinformation. Mithilfe dieses Frameworks können konzeptionelle, zeitliche und räumliche Eventrelationen intuitiv definiert werden. Zusätzlich werden Hierarchien für alle drei Dimensionen berücksichtigt, wobei die Hierarchien gewöhnlich als nützliches Hintergrundwissen zur Verfügung stehen. Mithilfe solcher Hierarchien können Muster auf verschiedenen Abstraktionsebenen und verschiedenen Granularitäten erkennt werden. Basierend auf diesem Framework adressieren wir die folgenden drei Probleme: (i) die Extraktion intervallbasierter Eventsequenzmuster, (ii) die Extraktion periodischer Eventmuster sowie (iii) die Extraktion semantischer Annotationen für die Ortsinformationen zu Events. Allgemein gesprochen betreffen die ersten beiden Probleme Eventkorrelationen, während das dritte Problem Korrelationen von Eventkomponenten betrachtet.

Insbesondere das erste Problem ist eine Verallgemeinerung der Erkennung sequenzieller Muster in traditionellen Datensätzen, in denen Muster komplexe zeitliche Beziehung zwischen Events auf verschiedenen Abstraktionsebenen und verschiedener Granularitäten darstellen. Das zweite Problem konzentriert sich auf die Suche nach periodischen Mustern. Hierzu werden vereinfachte Periodizitäten für Events, aber auch für Mengen von Events, die gemeinsam auftreten, zu Grunde gelegt. Das dritte Problem befasst sich mit der Extraktion von semantischen Annotationen für Orte, wobei Korrelationen von Zeit, Ort, und Kontext von Events ausgenutzt werden. Zur

Lösung der oben dargestellten Probleme stellen wir neue und effiziente Methoden vor. Unsere Experimente belegen klar, dass die gewonnenen Muster und das gewonnene Wissen für zahlreiche Anwendungen nützlich sind, beispielsweise für die Vorhersage von Events, für die semantische Suche nach Orten, oder einem Themen-basierte Clustering von Orten.

Acknowledgements

The accomplishment of this thesis would not have been possible without the support of many people. First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Dr. Michael Gertz, for his patient guidance and persistent support throughout my PhD study. I extremely appreciate all his time, insightful suggestions, and valuable comments to make this work possible. Moreover, I have learned from him not only the right way to conduct research but also how to think creatively and critically. Specially, his encouragement of thinking "out of the box" has definitely helped me a lot to accomplish the goal. I feel very lucky for the opportunity to be one of his PhD students.

Besides my supervisor, I would also like to thank the other committee members for their time and feedback. Their constructive comments have been definitely useful for me to clarify and refine the thesis.

I would like to acknowledge all my friends and colleagues for their interaction and support, professionally as well as personally, during my four years of study. The Database Systems Research Group at Heidelberg University has been a source of friendships as well as good advice and collaboration. I want to thank Ayser, Canh, Florian, Hamed, Christian, Jannik, and other members of the group for their constructive discussions and enthusiastic help. It has been a great time for me to work with those people. I also thank Chan, Nhu, and other friends for taking time to proofread and give useful comments during the preparation of this manuscript.

I am also thankful to the Ministry of Education and Training (MOET) Vietnam and Der Deutsche Akademische Austauschdienst (DAAD) for their financial support granted through their scholarship program.

Last but not least, I am deeply grateful to my parents, younger sister, and younger brother for their constant love, concern, and support. I could achieve nothing without the support of my family.

To my parents.

Contents

Chapter 1

Introduction

We are living in the "Age of Big Data", where the amount of data in our world has been exploding. Analyzing such data to extract insights or knowledge is a fundamental research area, popularly known as *Knowledge Discovery in Databases* or *Data Mining*, with many important applications to business intelligence, scientific exploration, e-government, medicine, or education [43]. Consequently, various data mining tasks have been addressed and realized for different applications, different analysis requirements from different users, as well as different types of data. Basically, these tasks differ in their goals, that is, the types of extracted *knowledge* they find.

Pattern mining [78] is one of the most important data mining tasks, where a *pattern* ('knowledge' in this context) is a substructure, such as a subset, a subsequence, or a subgraph, that significantly occurs in the analyzed data. For example, a pattern can be a set of products frequently purchased together by customers, a sequence of places most visited by tourists, or groups of social network users that share the same interest. Generally speaking, pattern mining is to find '*interesting*' patterns from a given dataset, where each pattern is formulated on the basis of a predefined *pattern language* whereas the interestingness of that pattern is determined by a set of predefined *constraints*. For different applications, such a pattern language as well as pattern constraints can be defined in different ways.

A famous instance of pattern mining is *frequent itemset mining* [1], where a pattern is defined as a set of items that often appear together in a transaction, and the interestingness of a pattern is specified based on a frequency-based constraint. For example, from customer transactions of an e-commerce company, one typically finds many items that are frequently purchased together, e.g., iPad and tablet cover, digital camera and memory card, or laptop and software. In other application domains, patterns representing more complex substructures can be found. For example,

sequential patterns are often discovered from DNA sequences [116], GPS trajectories [16], or Web usage data [15]. Frequent subgraphs might be found in protein-protein interaction networks [102], chemical compound structure data [123], or online social networks [30]. A co-location pattern representing a group of features of objects where objects exhibiting these features frequently co-occur is typically discovered from spatio-temporal data [51].

Many approaches to the discovery of patterns as mentioned above have been proposed and successfully applied in practice, for example, in identifying buying behaviors of customers for product and service recommendation, suggesting hypotheses for scientists, or extracting regularities and trends from historical data for future prediction. However, with the appearance and rapid growth of non-traditional data, especially *social media data*, finding interesting, useful patterns from such data poses many new challenging research issues.

With the rapid rise of the popularity of social media channels and near ubiquitous mobile access, massive amounts of data have been increasingly created and exchanged by users on the Internet. For example, people share their check-in locations on social networks (e.g., Facebook, Twitter, or Foursquare), provide information about events such as concerts or sports and share these events with other people (e.g., Last.fm, Eventful.com, or Eventbrite.com), contribute articles to collaborative projects (e.g., Wikipedia), or share their GPS trajectories (e.g., OpenStreetMap). Although a single data record of a social media source can be noisy, unreliable, and sometimes inaccurate, valuable knowledge can reliably be derived from a large dataset consisting of a significant number of data records. Additionally, with the rapid development of the Linked Open Data (LOD) cloud [46], a large variety of such data is widely available in the Resource Description Framework[1] (RDF), a standardized, machine processable form, e.g., DBPedia [11], LinkedGeoData [105], YAGO2 [48], or EventMedia [109]. Accordingly, social media data become potential data sources to extract valuable knowledge for many applications, such as online social network marketing, service recommendation, sales promotion, or enrichment of existing data sources.

In this thesis, we focus on the discovery of interesting, useful patterns from *event data*, where each event corresponds to "*something that happens at a given place and time*" and information about it is provided and shared on social media channels. For example, an event might be a public activity (e.g., a concert, festival, or sports competition), a historical event (e.g., a war, conflict, or battle), or a meteorological phenomenon (e.g., an observable weather event). Such data can be obtained from

[1] http://www.w3.org/RDF/

various sources, including popular social media channels sharing information about events, Wikipedia, or LOD sources.

Basically, a description of an event represented in a dataset contains three main components: (1) a context describing what that event is about, (2) a time component describing when that event occurred/will occur, and (3) a location component describing where that event occurred/will occur. For example, information about the context, time, and location of an event called "*Berlinale 2013*" are described as: "*the 63rd Annual Berlin International Film Festival*", "*from 7 to 17 February 2013*", and "*Berlin, Germany*", respectively. Such information about events can be provided as either raw textual data crawled from popular social media channels or standardized RDF-based datasets from the LOD cloud. Although event data in these sources are typically less noisy and more accurate than other types of social media data such as check-in data or microblogs, there are still challenges in fully exploiting such information. These challenges will be discussed in the next section.

The rest of this introductory chapter is organized as follows. First, Section 1.1 motivates and discusses the challenges related to the discovery of interesting, useful patterns from event data. We also outline our approaches to tackle these challenges. Next, the contributions of this thesis are summarized in Section 1.2. Finally, Section 1.3 outlines the structure of the thesis.

1.1 Problems, Challenges, and Our Approaches

For mining interesting, useful patterns from event data, we first discuss the necessary features of a comprehensive framework to model events in Section 1.1.1. Based on this framework, we then introduce our approaches to different pattern mining problems in Section 1.1.2.

1.1.1 A Unified Event Model for Pattern Mining Tasks

Similar to other types of social media data, a huge amount of event data has been increasingly generated from many social media sites. These data sources provide different information about events in terms of the number of attributes, data types, and even the meaning of attributes. Moreover, some event data sources are presented in the form of LOD sources employing different event ontologies, e.g., LODE [101], YAGO2 [48], or SEM [112]. The existence of heterogeneous data sources of events

motivates a general, comprehensive framework to model events for pattern mining purposes. Such a framework needs to address the following problems:

- Since there are various definitions for the term '*event*', one needs to precisely specify what an event is and which attributes describe an event in the model.
- Basically, an event pattern represents significant correlations among events, which are typically determined by exploiting conceptual, temporal, and spatial relationships between events. Modeling events needs to retain these relationships for pattern mining tasks.
- As useful background knowledge, concept hierarchies play an important role in deriving patterns at different levels of abstraction. In the context of event data, such hierarchies naturally exist for topics, time, and locations of events. For example, the topic of the event "*Berlinale 2013*" mentioned earlier can be generalized to topics "*International Film Festival*", "*Film festival*", "*Festival in Berlin*", or "*Festival*", based on, e.g., the Wikipedia categorization. Interesting patterns might consist of any of these topics. It is therefore necessary to support concept hierarchies in the framework.
- As a type of spatio-temporal data, location and time components of an event might be specified at different granularities, such as hour, day, or month for an event time, and address, city, or country for an event location. For example, a concert is often scheduled with a specific day and time and located at a particular address whereas a festival often occurs during a time interval and at different locations of a city. Thus, the framework needs to support multiple granularities for locations and time of events.
- The interestingness of a pattern is often subjective and varies from user to user. Moreover, in practice, the user might be interested in patterns that are valid in a specific time interval, in a specific geographic region, and/or simply refer to a specific concept. These requirements are typically specified as event constraints supported by the framework.

To the best of our knowledge, none of the existing approaches to the discovery of event patterns fully considers all these features as discussed above. For example, some approaches do not consider concept hierarchies, e.g., [53, 64, 73, 117, 124], whereas some others, e.g., [57, 58, 91, 92, 120], do not take into account geographic relationships between events. These approaches seem ad-hoc, because they are proposed for specific types of event data only.

Aiming at a comprehensive event model for the purpose of mining event patterns and knowledge discovery, we address the above problems and propose a framework to model events in Chapter 3. As shown later on, this framework is employed in all approaches presented in this thesis, where each approach is designed for a specific mining task.

1.1.2 Pattern Languages and Interestingness Measures

Since pattern mining is an application-oriented task, a pattern language and an interestingness measure are often specified for a particular application. In this thesis, we focus on three main research problems: (1) mining interval-based event sequence patterns, (2) mining periodic event patterns, and (3) extracting semantic annotations for locations of events. Although they employ the same unified event model as described in the previous section to extract useful knowledge from event data, they are different in terms of their objectives. We briefly describe these problems, including their challenges, and our respective approaches as follows.

- **Mining Interval-based Event Sequence Patterns**: Occurrences of real-life events often obey patterns. For example, a data mining conference frequently follows some workshop sessions; a burst of flu-related queries submitted to an on-line search engine appears before or during a flu outbreak; people are likely to visit nearby tourist attractions while attending a particular annual festival or cultural event. Finding patterns that exhibit significant correlations among events plays an important role for various applications such as event prediction, business intelligence, or user behavior analysis. However, this task is not trivial for the following reasons.

  First, one needs to define a pattern language that is powerful enough to express not only timepoint-based but also interval-based temporal relationships, such as *before*, *overlaps*, or *during*, between events. Moreover, most of the existing approaches to the problem of mining sequential patterns, e.g., [53, 89, 131], only allow one type of temporal relationships in a pattern. For example, a pattern "*a concert is followed by a cultural event, and this cultural event is followed by a social event*" is typically described as $concert \rightarrow cultural\_event \rightarrow social\_event$. However, that problem becomes more complicated if two or more types of temporal relationships appear in a single pattern. For example, how to model a pattern "*a music concert is performed after a cultural event and both of them occur during a festival*" so that it can be discovered by an algorithmic approach?

Next, a suitable interestingness measure is important to distinguish useful, interesting patterns from trivial or obvious ones. In addition, approaches to the discovery of patterns from event data have to take hierarchies associated with event components into account since interesting patterns might be obtained at any level of granularity and abstraction. Finally, one needs an efficient algorithm to discover such patterns from an event dataset, typically consisting of a significant number of events and associated with large concept hierarchies.

For the above challenges, we propose an approach to the discovery of sequential patterns from event data in Chapter 4.

- **Mining Periodic Event Patterns**: Periodicities commonly exist in real-life event data, e.g., daily activities of an individual, weekly concerts, monthly meetings, or annual festivals. Finding patterns that exhibit such periodicities is important for tasks such as predicting future events, analyzing user behavior for service suggestion, or identifying anomalies in data for fraud prevention. Approaches to the discovery periodic patterns from event data need to address the following problems.

  First, one needs a relaxed definition of periodicity since real-life events rarely obey perfect periodic patterns. For example, a particular concert is performed every Sunday, but sometimes, it might be canceled or moved to another day. Another problem is that sequences of events are typically not given but built by exploiting conceptual, spatial, and temporal relationships between events. Accordingly, traditional approaches to the discovery of periodic patterns, e.g.,[18, 42, 75], cannot be directly applied to event data. Similar to the previous pattern mining task, mining periodic patterns from event data also needs to take hierarchies associated with event components into account to derive patterns at different levels of granularity and abstraction.

  We will detail our approach that addresses the above problems in Chapter 5.

- **Extracting Semantic Annotations for Event Locations**: Meaningful, descriptive information about locations is essential for location-based services, e.g., location recommendation, location-based mobile advertising, or social event recommendation [29, 47, 93]. Unfortunately, such information tends to be poor in many data sources. Therefore, in Chapter 6, we propose a novel approach to extract semantic annotations for locations from event data. For this, we address the following problems.

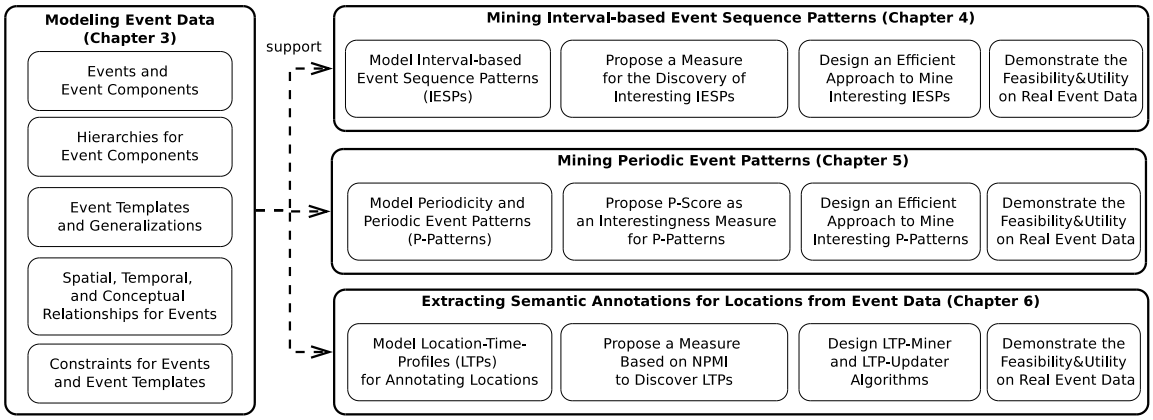


Figure 1.1: Overview of the major contributions of the thesis.

Given an event dataset, the key idea of annotating locations is to extract characteristic information of these locations from that dataset. However, this poses the first challenge: "What information from the event data is useful in semantically annotating locations?". Following this, another challenge arises: "How to efficiently extract this information?". Finally, assuming that annotations of locations are extracted, one needs a systematic approach to measure how good the extracted annotations are. This is also a big challenge since there is typically no pre-existing ground-truth for location annotations.

Our approaches to the problems mentioned above make contributions that are listed in the following section.

1.2 Contributions

Figure 1.1 illustrates the major contributions of this thesis. First, we propose a comprehensive framework to model events in Chapter 3. Including all the features listed in Section 1.1.1, this framework provides the basis for our approaches to the three research problems of mining event patterns discussed in Section 1.1.2. These approaches will be detailed in Chapters 4, 5, and 6. To summarize, this thesis makes the following contributions:

- **A Comprehensive Framework to Model Events**: We propose a comprehensive framework to model events, where each event is described by three components: context, time, and location. This allows one to flexibly define how events are related in terms of conceptual, temporal, and spatial relationships. In addition, these event components can be individually generalized to higher

levels of abstraction and granularity to derive *event templates* describing topics of events. The concepts and notions of events and event templates are fundamental to define a pattern language and an interestingness measure for a specific pattern mining approach later on. Finally, to guide the search for interesting patterns, this framework also supports the user to specify constraints on events and event templates.

- **An Efficient Approach to the Discovery of Interval-based Event Sequence Patterns**: Based on the above framework, we address the problem of mining interval-based event sequence patterns. Different from traditional approaches that consider only timepoint-based relationships between event occurrences, we introduce a pattern specification language that is able to express complex temporal relationships for a set of interval-based events. For example, a pattern "*a music concert is performed after a cultural event and both of them occur during a festival*" can naturally be described with our pattern language. Furthermore, we propose an interestingness measure and present an algorithmic approach to efficiently extract complex patterns at different levels of abstraction and granularity based on hierarchies. We demonstrate the feasibility and utility of our framework by using two different real datasets from YAGO2 and the Website eventful.com.

- **An Efficient Approach to the Discovery of Periodic Patterns**: Here we consider the problem of mining interesting periodic patterns in the presence of hierarchies for contexts, time, and locations of events. We utilize the event model introduced earlier to formulate periodic event patterns, which describe relaxed periodicities of event topics. For this, we introduce a probabilistic measure called *p-score* to determine how likely an event topic occurs (or multiple event topics co-occur) during a time window. We propose an effective algorithm for the discovery of significant periodic event patterns from a dataset of events. Finally, we demonstrate the feasibility and utility of our framework by using datasets of real-life events, e.g., concerts, sports, or festivals.

- **An Event-based Framework to Extract Semantic Annotations for Locations**: We focus on the problem of enriching location information on the basis of correlations among event topics, locations, and time. For this, we formulate Location-Time-Profiles (LTPs) and propose a measure based on Pointwise Mutual Information to extract LTPs consisting of event topics that describe characteristics of a particular location. For example, considering event topics

related to festivals, '*Oktoberfest*' and '*Beer*' are related to the city of Munich (Germany) whereas '*Opera*' and '*Puccini festival*' are related to the city of Torre del Lago (Italy). We also present an efficient algorithm to extract semantic annotations for locations from event data. Such information of locations can be exploited further for tasks such as topic-based search for locations or clustering locations based on their semantic similarity. Since the number of locations for annotating is often large, it is infeasible to manually validate the result. We therefore propose an indirect method to evaluate how good the extracted annotations are with hierarchical clustering. Additionally, taxonomies of locations are then built from the analysis of location clusters. To deal with periodic updates of event datasets, we furthermore give a scalable and efficient approach to incrementally update location annotations. Finally, we use real event datasets to demonstrate the performance of our approach.

1.3 Thesis Organization

The rest of this thesis is organized as follows.

In Chapter 2, we give a general introduction to the pattern mining problem. We also discuss existing approaches to the discovery of patterns from spatial/temporal data as well as existing approaches that utilize concept hierarchies to find patterns. In Chapter 3, we describe a comprehensive framework to model events. Based on that, we propose approaches to the discovery of interval-based event sequence patterns, periodic event patterns, and semantic annotations for locations from event data in Chapters 4, 5, and 6, respectively. Finally, we summarize the thesis and discuss some interesting directions for future studies in Chapter 7.

Chapter 2

Background and Related Work

In this chapter, we first give a general introduction to the pattern mining problem. We then discuss approaches related to the pattern mining problem in the context of spatial and temporal data. Finally, we review pattern mining approaches exploiting concept hierarchies since they are also related to our work.

2.1 The Pattern Mining Problem

In this section, we consider a theoretical perspective of the *Pattern Mining Problem*, which is based on concepts and notations as introduced by Mannila and Toivonen [78].

Given a dataset $\mathcal{D}$, a predefined pattern language $\mathcal{L}$ for expressing patterns, and a predicate ϕ for evaluating whether a pattern $\varphi \in \mathcal{L}$ is '*interesting*' or not, the set of all interesting patterns with respect to $\mathcal{L}$, $\mathcal{D}$, and ϕ is defined as

$$Th(\mathcal{L}, \mathcal{D}, \phi) = \{\varphi \in \mathcal{L} \mid \phi(\mathcal{D}, \varphi) \textit{ is true}\}. \tag{2.1}$$

Based on this definition, the pattern mining problem is defined as to efficiently compute the set $Th(\mathcal{L}, \mathcal{D}, \phi)$, i.e., finding the theory of $\mathcal{D}$ with respect to $\mathcal{L}$ and ϕ.

Generally speaking, a pattern is a substructure of data, e.g., an itemset, a subsequence, or a subgraph. A set $\mathcal{L}$ of patterns with the same structural characteristic (e.g., set, sequence, or graph) is called a pattern language, which is application-specific. The interestingness of a pattern is determined on the basis of a predefined predicate ϕ, which is typically a single pattern constraint or a conjunction of several pattern constraints. For example, in the *frequent itemset mining* problem [1, 43], a pattern is defined as a set of items that often appear together in a transaction, and the interestingness of a pattern φ is determined by $\phi(\mathcal{D}, \varphi) := (Support(\mathcal{D}, \varphi) \geq$

min_support), where $Support(\mathcal{D}, \varphi)$ is the number of transactions in the dataset $\mathcal{D}$ that contain all items of the pattern φ, and *min_support* is a user-defined threshold.

Based on Equation (2.1), it is straightforward to compute the set of all interesting patterns by iterating through every element $\varphi \in \mathcal{L}$ and evaluate $\phi(\mathcal{D}, \varphi)$. However, this naive approach is clearly inefficient and sometimes infeasible. In particular, if the set $\mathcal{L}$ is infinite, then this approach will not terminate. Therefore, computing the set $Th(\mathcal{L}, \mathcal{D}, \phi)$ without exploring the (infinite) set $\mathcal{L}$ is often considered. Such an approach is typically based on certain characteristics of the set $\mathcal{L}$ and the predicate ϕ. That is, a partial order $\preceq$, called specialization relation, is assumed over the set $\mathcal{L}$, and the predicate ϕ is *monotonic/anti-monotonic* with respect to $\preceq$. The definitions of monotonicity and anti-monotonicity properties are as follows.

Definition 2.1 (Monotonicity) *Let $\mathcal{D}$ be a dataset and $\mathcal{L}$ be a set of patterns with a partial order $\preceq$. A predicate ϕ is* ***monotonic*** *with respect to $\preceq$ iff $\forall \varphi, \psi \in \mathcal{L}$, $\varphi \preceq \psi$ and $\phi(\mathcal{D}, \varphi)$ is true $\Rightarrow \phi(\mathcal{D}, \psi)$ is true.*

Definition 2.2 (Anti-monotonicity) *Let $\mathcal{D}$ be a dataset and $\mathcal{L}$ be a set of patterns with a partial order $\preceq$. A predicate ϕ is* ***anti-monotonic*** *with respect to $\preceq$ iff $\forall \varphi, \psi \in \mathcal{L}$, $\varphi \preceq \psi$ and $\phi(\mathcal{D}, \varphi)$ is false $\Rightarrow \phi(\mathcal{D}, \psi)$ is false.*

For example, in the context of the frequent itemset mining problem mentioned above, the support constraint $\phi(\mathcal{D}, \varphi) := (Support(\mathcal{D}, \varphi) \geq min\_support)$ is anti-monotonic over itemsets with respect to the subset relation $\subseteq$. This is commonly known as the *Apriori property*, stated as "*all nonempty subsets of a frequent itemset must also be frequent*" [43]. In other words, if an itemset is infrequent then every superset of it is also infrequent.

The monotonicity (or anti-monotonicity) property is important to develop an efficient approach to compute the set $Th(\mathcal{L}, \mathcal{D}, \phi)$ without exploring the whole set $\mathcal{L}$. Such an approach is based on the following strategies:

(1) the patterns of the set $\mathcal{L}$ are considered in the order of the most generalized patterns to the most specialized ones (vice-versa for the case of anti-monotonicity), and

(2) if a pattern φ is not interesting (i.e., $\phi(\mathcal{D}, \varphi)$ is false) then all elements of the set $\mathcal{L}$ that are specializations of the pattern φ can be safely pruned.

These strategies allow developing efficient algorithms that utilize pattern space pruning techniques for a specific pattern mining problem, such as the *Apriori* algorithm [1, 43] for mining frequent itemsets, or various *Apriori-like* algorithms for

mining sequential patterns, e.g., [2, 104, 131]. As shown in Chapters 4 and 5, we also employ these strategies to efficiently mine interesting patterns from event data.

Since pattern mining is an application-driven research problem, several pattern mining approaches have been proposed for several applications and data types. Although these approaches are typically based on the theoretical, general model as described above, they differ in the types of patterns (pattern languages) as well as interestingness constraints to deal with, for instance, mining sequential/periodic patterns from time-related data or mining co-location/co-occurrence patterns from spatial/spatio-temporal data. In the following, we review important approaches to the pattern mining problem in the context of spatial, temporal, and spatio-temporal data, which are closely related to our work.

2.2 Mining Patterns from Spatial, Temporal, and Spatio-Temporal Data

With modern data acquisition technologies such as remote sensors, GPS tracking systems, or mobile devices, huge amounts of spatial and spatio-temporal data (ST-data), such as sensor data, trajectories of moving objects, event data, or data in sports analytics, have increasingly been collected. The success of substantial research in spatial and temporal data models and database systems gives rise to new approaches for extracting useful patterns from such data, see, e.g., [31, 32, 40, 77, 97]. Many types of patterns that are typically found in transaction data have been formulated in the context of ST-data [81]. For example, co-location patterns [51], where each pattern is defined as a set of spatial features that often co-occur, are equivalent to frequent itemsets. Sequential and periodic patterns typically found in transaction data also exist in trajectory data and event data, see, e.g., [53, 65, 66, 71].

Recent approaches show that patterns discovered from ST-data can be utilized as well in various application domains. For example, co-location and co-occurrence patterns are useful in understanding ecological predator-prey relationships, in transportation planning, or in identifying tactics in battlefields, sports, and games [20]. Patterns extracted from user activity data (e.g., mobile phone usages, user trajectories, or check-in data) can infer social relationships of individuals and the behavior of users [64, 72, 84]. In the computer network security domain, patterns describing spatial and temporal correlations among network events are valuable for understanding global network phenomena, such as fault cascading in communication networks [117].

Analyzing spatial and temporal trends of keywords from search engine query data can improve early detection of infectious diseases, e.g., an approach employed by Google to detect influenza epidemics [39].

Although many efficient approaches to the discovery of useful patterns from transaction data have been proposed, utilizing these approaches to fully exploit ST-data for useful patterns has many challenges. For example, compared to transaction data, mining ST-data needs to take spatial and temporal relationships into account. Approaches based on partitioning space/time for transactions might not be natural due to the continuity of space and time [53]. In particular, spatial/temporal relationships among objects across transactions might be lost when partitioning space/time for transactions.

In the following subsections, we review key approaches related to the discovery of patterns from ST-data, which are closely related to our work. Section 2.2.1 describes prominent approaches related to co-location and co-occurrence patterns. Sections 2.2.2 and 2.2.3 discuss approaches to the discovery of sequential and periodic patterns, respectively, where temporal relationships play an important role in building patterns.

2.2.1 Co-location Pattern Mining

In the context of spatial/spatio-temporal pattern mining, *features* (or object-types) are key ingredients in constituting patterns, analogous to the concept of '*item*' in the frequent itemset mining problem. Some examples of features include species of plants or animals, location types (e.g., hotel, restaurant, or parking lot), road types, crime, and diseases.

Given a set F of features, a (spatial) co-location patterns is a subset of F containing features that frequently appear nearby together in some geographic space. For example, as illustrated in Figure 2.1, the set {'gas', 'fast food'} is an example of a co-location pattern describing that a gas station and a fast food restaurant are often located together. Nile Crocodile and Egyptian Plover Bird constitute another prominent example of a co-location pattern [51]. Note that in the problem of mining co-location patterns, *prevalent* patterns mean *interesting* patterns. Existing approaches to the discovery of interesting co-location patterns can be categorized into three groups: *statistics-based*, *clustering-based*, and *association rule-based* approaches. We review work in each group in the following.

Statistics-based approaches use statistical measures to estimate spatial correlations between features, such as the cross-K function [27] or G-function [37]. A com-

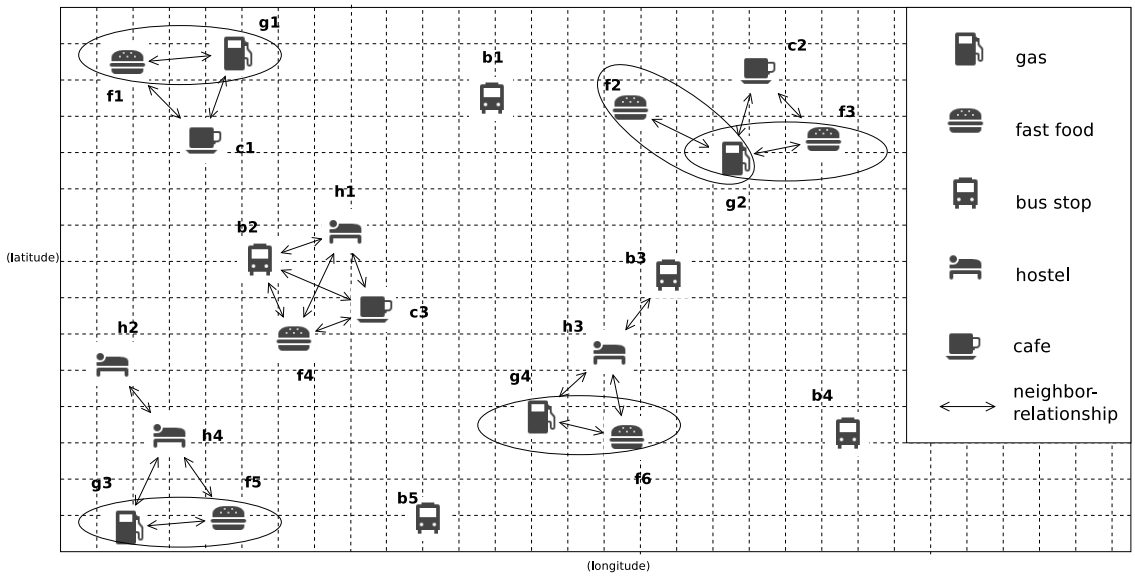


Figure 2.1: Example spatial dataset. Instances of each feature in the set {'gas', 'fast food', 'bus stop', 'hostel', 'parking', 'cafe'} are placed at different geographic locations. An instance of the feature 'gas' and an instance of the feature 'fast food' tend to be located nearby.

mon limitation of these approaches is that computing all possible co-location patterns is computationally expensive. Such approaches are therefore not scalable. Only few works belonging to this category have been proposed.

Traditional data clustering techniques can be utilized to discover co-location patterns. Clustering-based approaches are further divided into two subcategories: *object-clustering* and *feature-clustering*. Castrol et al. [33, 34] use an object-clustering approach to find prevalent co-location patterns. For each feature, objects exhibiting this feature are clustered. Boundaries of clusters are also computed. Based on the Apriori algorithm [43], candidates of co-location patterns are generated. From these, interesting ones are obtained based on a prevalence measure computed from the overlapping areas of respective cluster boundaries. Since this prevalence measure is anti-monotonic, the above approach can utilize the strategies as discussed in the previous section to efficiently prune the pattern space. However, the quality of results heavily depends on the quality of clustering processes. Thus, such an object-clustering approach fails in cases where objects of one or more features cannot be clustered well.

Instead of clustering objects, Huang et al. [54] cluster features to find co-location patterns. The authors treat each feature as a layer and cluster these layers. They define a density-based metric to compute the similarity between two layers. Basically, this metric is computed from the area of overlapping regions of respective layers.

Discovered clusters are considered co-location patterns. The main drawback of this method is that each feature is assigned to only one cluster. Consequently, one feature occurs in at most one co-location pattern. This is not appropriate in many application domains. Furthermore, the quality of results is impacted by the selected clustering algorithm as well as the characteristics of the input data.

The third category consists of approaches motivated by frequent itemset mining approaches. The Apriori Algorithm [43] is suitably applied for mining co-location patterns. Following this, there are two main directions: *transaction-based* and *instance-based* approaches.

The key idea of a transaction-based approach is to define transactions for the input dataset so that the Apriori algorithm can be applied. Koperski and Han [61] propose a method to define transactions based on a set of predefined predicates, e.g., is_a(X, *house*) or close_to(X, *beach*). A transaction consisting of predicates is then generated for each object belonging to a *reference-feature* specified by the user. For example, {is_a(X, *house*), close_to(X, *beach*), is_expensive(X)} is a transaction corresponding to the object X of the reference-feature *house*. Prevalent patterns discovered by applying the Apriori algorithm are relevant to the selected reference-feature. Generalizing this approach to the case where no reference-feature is specified is nontrivial.

Different from transaction-based approaches, instance-based approaches do not rely on defining transactions. Instead, instances of patterns are computed dynamically based on spatial distances between objects exhibiting the relevant features. Such instances are then utilized to compute the interestingness (here, the prevalence) of patterns. Three common steps are: (1) candidate pattern generation, (2) instance collection, and (3) prevalence evaluation. Most of the recent approaches use the Apriori algorithm in the first step to generate candidates. However, they differ in implementing and optimizing the last two steps.

Morimoto [83] enumerates instances for a co-location pattern based on space partitioning. In his approach, the interestingness of a co-location pattern is simply the number of the corresponding instances. However, to obtain the anti-monotonic property for this measure, the author assumes a non-overlapping instance constraint, that is, any object must belong to only one instance. Thus, this approach might find an incomplete set of patterns because of missing instances [51].

To obtain completeness and correctness in mining co-location patterns, Huang et al. [51] define another prevalence measure, called the *participation index*, which not only allows overlapping instances but also has the anti-monotonic property. Their approach, known as a full-join approach, is based on joining instances of size-k pat-

terns to compute instances of size-(k+1) patterns. For example, instances of the pattern {'gas', 'fast food', 'cafe'} are computed based on instances of {'gas', 'fast food'} and {'gas', 'cafe'}. This method is inefficient for dense datasets containing a large number of instances. Therefore, recent approaches have been proposed to improve the performance of the mining process, including partial-join approach [129], joinless approach [130] and density-based approach [122].

Yoo et al. [129] propose a partial-join approach based on finding maximal cliques to reduce the number of join operations in generating instances. However, finding maximal cliques is also computationally expensive. Thus, the authors also propose a joinless approach [130] based on an instance-lookup scheme. Instead of finding maximal cliques, they build a table of so-called *star instances* to enumerate instances. A disadvantage of this approach is the memory cost to store the table of star-instances.

Another direction to reduce the number of join operations in enumerating instances is based on an upper bound of the prevalence measure. Xiao et al. [122] propose a method to early prune false pattern candidates based on dividing space into partitions, for example, cells of a grid. In the step of generating size-k instances, these partitions are sorted by descending density of size-(k-1) instances. Instead of considering all objects in the whole space, only objects of a partition are considered to enumerate instances. Based on this, an upper bound of the prevalence is computed for each pattern candidate, and false pattern candidates can be pruned early.

Note that all approaches mentioned above consider only the spatial dimension for mining co-location patterns. For spatio-temporal data, co-location patterns can be extended to co-occurrence patterns to represent features that frequently occur nearby in space and time, e.g., [20, 115, 124]. Although these approaches take the temporal dimension into account, they only consider temporal relationships between objects to enumerate instances, i.e., at the instance level. Thus, their pattern language cannot express temporal relationships, e.g., *before*, *after*, or *follows*, between features. In the next section, we review well-known approaches to the discorvery of sequential patterns where temporal relationships between features are taken into account.

2.2.2 Sequential Pattern Mining

Whereas co-location/co-occurrence patterns are subsets of object features, a sequential pattern is typically described as a sequence of features where the order of features in this sequence is important. Sequential patterns are often extracted from sequence data, e.g., customer purchasing records, stock data, web log of a click stream, or

DNA sequences. Approaches to the discovery of sequential patterns are reviewed in the following.

The problem of mining sequential patterns has been introduced by Agrawal and Srikant [2] for the purpose of extracting all frequent subsequences from time-related data. In their approach, a sequence is a list of transactions ordered by transaction time, and a transaction is a set of items. They propose an algorithm, called *AprioriAll*, that employs the Apriori property, i.e., any super-pattern of an infrequent pattern cannot be frequent. To improve the performance of *AprioriAll*, the authors present a new algorithm called *GSP*, which employs a hash structure for fast support counting in their subsequent work [104]. The algorithms *GSP* and *AprioriAll* create a large number of pattern candidates and also make multiple passes over the database. In other words, these approaches are not efficient.

For the purpose of improving the sequential pattern mining performance, several approaches have been proposed. Zaki [131] proposes a lattice-based approach, called SPADE (Sequential PAttern Discovery using Equivalence classes), to minimize the computational cost. In addition, this approach can discover frequent subsequences of subsets of items (e.g., $\{a, b\} \rightarrow \{c\} \rightarrow \{a, d, e\}$), not just subsequences of single items (e.g., $a \rightarrow c \rightarrow b$) as in the approach of Agrawal et al. [2]. In the SPADE approach, a vertical layout of the sequence database, called *id-list*, is built for fast instance counting and also reducing the number of database scans. Based on the *id-list*, a lattice of patterns is constructed on-the-fly with an initialization based on size-1 and size-2 frequent subsequences, which are precomputed using an Apriori-like method. To solve the problem that the lattice may not fit in main memory, this approach partitions the lattice into disjoint subsets, where each subset, called an *equivalence class*, can be separately processed in main memory. The lattice can be traversed in either breadth-first (BFS) or depth-first search (DFS). BFS takes advantage of more information available for pruning since the frequent patterns at the current level (size) must be completely discovered before processing the next level. On the other hand, DFS requires less memory than BFS, and therefore, it may be the only feasible approach when the number of frequent patterns is large.

Similar to SPADE, another approach, called SPAM [8], employs a data structure that is assumed to fit in main memory, called a vertical bitmap, for efficient support counting. This approach also assumes a transaction database as input. Each item appearing in transactions of the database is represented by a bitmap, where the *i-th* bit of the bitmap corresponds to the presence or absence of the item in the *i-th* transaction. A pattern tree is constructed as follows. The root of the tree is an empty

sequence. The children of a node in the tree are created by extending the sequence corresponding to the node using either *sequence-extension* step (i.e., increasing the sequence length) or *itemset-extension* step (i.e., increasing the number of items for the itemsets of the current sequence). SPAM traverses the tree in depth-first-search and utilizes the vertical bitmaps to efficiently prune infrequent patterns. Experiments show that SPAM outperforms SPADE in terms of computation time, but it is less efficient than SPADE in terms of memory consumption [76].

An alternative to the *generate-and-test* method appearing in Apriori-based approaches is the *pattern-growth* method introduced in some approaches such as FreeSpan [44] or PrefixSpan [89]. Generally, these approaches generate as few pattern candidates as possible by growing longer patterns from shorter ones. Such approaches are typically based on *projected-databases* to partition the search space. Let α be a subsequence and $\mathcal{D}$ be a dataset of sequences. In the approach FreeSpan [44], sequences of the α-projected-database are derived from sequences of $\mathcal{D}$ that have α as a subsequence. In the approach PrefixSpan [89], the α-projected-database contains sequences where each one is a postfix of some sequence in $\mathcal{D}$ that has α as a prefix. For each pattern α, only the α-projected-database is considered to compute the support for α and to generate super-patterns of α. Experimental results show that PrefixSpan outperforms FreeSpan [89]. In the experiments of Ayres et al. [8], both PrefixSpan and FreeSpan do not perform well for large datasets.

To reduce the computation time and also to find only patterns that satisfy the requirements of the user, constraint-based approaches are often considered. Mannila et al. [79] propose an approach to mine frequent episodes in a sequence of events, where each episode follows some predefined template. A template is defined as a directed acyclic graph of event types. For example, $type\_A\_event \rightarrow type\_B\_event$ is an example of a serial episode template. Another direction to constrain types of patterns in the output is using *regular expressions*, such as an approach called SPIRIT [36]. Generally, the efficiency of a constraint-based approach depends on the properties of the constraints (i.e., monotonicity or anti-monotonicity). Such properties allow pushing constraints inside the pattern mining process so that false pattern candidates can be pruned early. However, some constraints are neither monotonic nor anti-monotonic (e.g., regular expression constraints). For such a constraint, a possible solution might be based on relaxing that constraint [36] or generalizing the monotonic/anti-monotonic properties, e.g., to *prefix-anti-monotonicity* [90].

All the approaches mentioned above consider only the temporal dimension, i.e., in terms of chronological order of transactions. Moreover, these approaches assume that

sequences explicitly exist in the input dataset. For example, $a \rightarrow \{b, c, d\} \rightarrow \{a, c\}$ is a sequence of 3 transactions, where the first transaction consists of an item a, the second transaction consists of items b, c, and d, and the last transaction consists of two items a and c. In the context of spatio-temporal data, such sequences are typically not given or simply do not make sense. Indeed, such sequences will be derived from both spatial and temporal relationships among events in the spatio-temporal dataset. Thus, new approaches come into play for the purpose of mining sequential patterns from spatio-temporal data.

One direction to tackle the problem in which sequences of events are not explicitly given is a method based on partitioning the space into discrete locations. For each location, events occurring at this location constitute a sequence. For example, an approach proposed by Tsoukatos and Gunopulos [110] uses a grid and a reverse-z order to enumerate sequences of environmental events, e.g., temperature or atmospheric pressure. Subsequently, sequential patterns can then be discovered by one of the approaches such as SPADE or PrefixSpan. Another direction is based on identifiers of objects producing events to build event sequences from moving object datasets [16, 17]. Generally, approaches in both directions are designed for *only specific* spatio-temporal data types, e.g., moving object data or environmental data. As mentioned earlier, partitioning space for transactions is not natural, and object identifiers might not exist in some cases.

Huang et al. [53] propose a general framework to discover sequential patterns from a dataset of events in which both spatial and temporal relationships between events are considered. Based on the concept of a *spatio-temporal neighborhood* of events, significant sequential event patterns can be extracted without explicitly defining event sequences. We also employ this idea to model event patterns and instances in Chapter 4. However, different from their approach where they consider only *follows*-relationships between time point-based events, our approach can discover event patterns expressing complex combinations of multiple temporal relationships, e.g., *before*, *during*, or *overlaps*, among interval-based events. Furthermore, with the presence of topic-based, spatial, and temporal hierarchies of events, our approach is able to derive spatio-temporal patterns at different levels of abstraction and granularity. This has not been considered by the approach of Huang et al. This idea will be detailed in Chapter 4.

2.2.3 Periodic Pattern Mining

Generally, *periodic patterns* describe regular appearances of some elements in a long sequence, such as a time series or a moving object trajectory. Different from mining sequential patterns as mentioned in the previous section, mining periodic patterns is to find patterns that occur in some regularity. For example, some data mining and machine learning conferences annually co-occur, or an individual tends to follow the same route from his home to his office every working day of the week. Finding such patterns is important for many applications, for instance, forecasting upcoming events, predicting user behavior for service suggestion, or for better understanding the inherent characteristics of the underlying dataset. In the following, we discuss problems and existing approaches related to mining periodic patterns from time-related data.

First, *periodicity* needs to be explicitly modeled. Given a sequence of elements (*features* as mentioned in Section 2.2.1), if an element periodically appears in that sequence, then it will be found in that sequence in every fixed time interval, called *cycle*. A *period* is a duration of one cycle that an element re-occurs. Most of the traditional approaches to the discovery of unknown periods from time series or sequence data are typically based on the *Discrete Fourier Transform* (DFT) and its variations [113]. Since in Chapter 5 we will employ one of these methods as a tool to detect periods for periodic patterns, we briefly describe the DFT in the context of periodicity analysis for time series data as follows.

Given a time series of N real values, denoted $x(n)$, $n = 0, 1, 2, ..., N-1$, the DFT of a sequence is a sequence of N complex numbers defined as:

$$X(k) := \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N} \qquad k = 0, 1, 2, ..., N-1. \tag{2.2}$$

Note that each $X(k)$ is a complex number that encodes the amplitude and phase of a sinusoidal component of the original data $x(n)$. The frequency, amplitude and phase of this sinusoid are $\frac{k}{N}$, $\frac{|X(k)|}{N}$, and $\arg(X(k))$, respectively. Based on $X(k)$, the original data $x(n)$ can be represented as

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{i2\pi kn/N} \qquad n = 0, 1, 2, ..., N-1. \tag{2.3}$$

The DFT of a given time series can be efficiently computed in $O(N \log(N))$ time, where N is the length of the time series, with a Fast Fourier Transform (FFT), e.g.,

the Cooley-Tukey algorithm [113]. Based on the DFT, the *power spectral density* at a frequency $\frac{k}{N}$ can be estimated as $PSD(k) = |X(k)|^2$. This value represents how much the expected signal power at that frequency is. Since period is the inverse of frequency, significant periods can easily be detected by selecting the frequencies carrying most of the energy. For sparse data, the Lomb-Scargle periodogram [74, 99], known as a variation of the Fourier transform, can improve the performance.

Another method to detect significant periods is using the circular AutoCorrelation Function (ACF) [114] defined as

$$ACF(t) = \frac{1}{N}\sum_{n=0}^{N-1} x(t).x(n+t) \qquad t = 0, 1, ..., N-1. \tag{2.4}$$

Generally, the value $ACF(t)$ represents how similar the sequence $x(n)$ is to its previous values for different t lags. The higher the $ACF(t)$ is, the more likely t is a true period. Thus, based on this, significant periods can be automatically identified.

For each period t that is detected by any of the methods mentioned above, a periodic pattern is then a length-t subsequence of the original sequence. However, such a pattern is only able to represent a *full-cycle* periodicity in the sense that all points in a cycle have to be repeated. In cases where only some points have a cyclic behavior, approaches to the discovery of *partial* periodic patterns [42, 45, 75] can be employed. For example, a music event that occurs every Monday and Saturday but might or might not occur on other days of the week can be represented as a pattern (music,*,*,*,*,music,*). A more flexible model of partial periodic patterns is a model of *asynchronous* periodic patterns [126]. An asynchronous periodic pattern is a subsequence of the original sequence that only repeats in some disjoint segments of the original sequence, and two consecutive segments are allowed to be separated by at most a time distance threshold. Some other variations of periodic patterns are *surprising* periodic patterns modeled by an information gain metric [125], partial periodic patterns with *gap penalties* [127], *cyclic association rules* [85], or *representative trends* based on relaxed periods [55].

In each approach mentioned above, periodic patterns are discovered from a long sequence that is assumed to obey some periodicity. Such an approach cannot directly be applied to spatio-temporal data for the following reasons. As mentioned in Section 2.2.2, sequences are not typically given in a spatio-temporal dataset (e.g., event data) but are often derived from spatial and temporal relationships between events. Although in the context of moving object data, an object trajectory can be considered a sequence of geo-coordinates, a coordinate rarely repeats itself exactly in such

a sequence [18, 77]. Therefore, new approaches are needed for spatio-temporal data. Since such approaches are closely related to our approach presented in Chapter 5, they will be discussed in more detail and compared with our work there.

2.3 Concept Hierarchies and Pattern Mining

As useful background knowledge, concept hierarchies play an important role in pattern mining approaches to the discovery of patterns at different levels of abstraction. Such hierarchies naturally exist in various real-world application domains, typically in the form of a taxonomy tree or a more complex structure such as a directed acyclic graph (DAG). In this section, we review recent approaches to the discovery of patterns at different levels of abstraction in the presence of concept hierarchies.

Srikant and Agrawal [103, 104] generalize the problem of mining frequent itemsets and association rules for cases where hierarchical relationships (*is-a* relationships) between items are available. Their approach aims at finding generalized association rules consisting of items at different levels of the underlying hierarchy. Since this approach uses a single support threshold for all levels of abstraction, many uninteresting rules will be included in the result if the threshold is rather low, but many interesting rules will be discarded if the threshold is rather high. Instead of using a uniform threshold for support, Han and Fu [41] propose an approach using multiple support thresholds, where each threshold corresponds to a particular level of the hierarchy. Consequently, each pattern discovered by this approach only consists of items at the same level of the hierarchy [92]. Moreover, explicitly specifying support thresholds based on hierarchy levels might not always be trivial, especially in cases where concept hierarchies are represented by DAGs.

Several approaches to the discovery of patterns from multi-dimensional databases can be considered related to this field. Pinto et al. [91] propose an approach to find sequential patterns in the presence of multi-dimensional information for sequence data. For example, multiple attributes of a customer (e.g., job, age-group, or income-group) can be embedded in his transaction sequence, such as $\langle$(*business*, *middle-aged*, *high-income*), $\langle$abacb$\rangle\rangle$, where $\langle$abacb$\rangle$ is a sequence of products purchased by that customer. This approach allows attributes to be generalized to a meta-symbol '*' (referred to ALL) in a pattern. For example, the pattern $\langle$(*, *, *high-income*), $\langle$ab$\rangle\rangle$ describes that customers belonging to the high-income group will buy item 'b' after buying item 'a'. Thus, this approach can be considered a special case of multilevel

pattern mining, where only two levels are considered, namely the most specific level and the most generalized one (denoted by '*').

A more general approach to the discovery of multilevel patterns from multidimensional databases is proposed by Plantevit et al. [92]. Similar to our approach that will be detailed in Chapter 4, their approach can deal with sequential patterns in which each item can be a tuple of multiple attributes. However, although their approach can handle concept hierarchies, their patterns are only able to represent timepoint-based relationships. Moreover, they do not consider spatial relationships and therefore their approach cannot be applied for spatio-temporal data.

In the Semantic Web, ontologies, often described within the Resource Description Framework (RDF), can be considered promising sources to obtain concept hierarchies. In such RDF-based data sources, an RDF statement is the basic element consisting of a subject, a predicate, and an object. Based on RDF, hierarchical relationships between concepts can be defined with special predicates, e.g., *is-a*, *sub-class-of*, or *type*. Jiang and Tan [57, 58] propose an approach based on traditional algorithms for mining frequent itemsets from transaction databases to discover frequent *relationsets* (sets of RDF statements) from RDF data. Using concept hierarchies embedded in RDF data, this approach can find patterns containing not only original RDF statements in the RDF dataset but also generalizations of such RDF statements. However, since this approach defines a transaction as an RDF document that contains RDF statements, it cannot be applied for spatio-temporal data. As mention earlier, in the context of spatio-temporal data, defining transactions is not natural due to the continuity of space and time.

In summary, pattern mining is an application-driven problem which is often addressed and realized for a particular application and data type. In this chapter, we reviewed prominent approaches that are closely related to our work. Briefly, none of these approaches can be applied to event data which are provided and shared by social media platforms. Before describing our approaches, we introduce a comprehensive framework to model events in the next chapter.

Chapter 3

Event Data and Event Model

For the purpose of data mining and knowledge discovery from event data, one needs a comprehensive framework to model *events*. With the explosive growth of the Linked Open Data cloud of datasets, various event ontologies have been proposed to describe event information. However, to the best of our knowledge, there is no model in existing work that fits the pattern mining tasks we focus on in this thesis well, even there are still various definitions for the term '*event*'. The importance of an event model for the pattern mining approaches described in the later chapters motivates us to conduct a systematic study of a comprehensive and flexible model.

In following section, we define the term '*event*' that will be used in this thesis. We then sketch an event model consisting of context, time, and location components. These three components will be described in detail in Section 3.2 and Section 3.3. Based on that, Section 3.4 formulates events and introduces the concept of event templates to describe event topics. Finally, we introduce event relationships and event constraints in Sections 3.5 and 3.6.

3.1 Preliminaries

Since there is no single agreed upon formal definition for the term '*event*', we first specify *what an event is* in our context. We then discuss the availability and heterogeneity of event data sources on the Internet. Finally, we propose an event model for the purpose of pattern mining and knowledge discovery, which is simple but flexible to deal with the heterogeneity of such event data.

3.1.1 What is an Event?

The term '*event*' is used in many domains and it has several meanings. In general, an event refers to an observable occurrence or a phenomenon in the real world. For example, an event might be a festival, a live concert, or a sports competition. In meteorology and climatology, meteorological phenomena are observable weather events such as hurricanes, blizzards, or floods. In social media data, a check-in, a tweet, or a comment of a user might also be considered an event. In software development, an event can be a user action (e.g., a keystroke or a mouse click) or a system occurrence (e.g., running out of memory or a firewall event). Moreover, an event might be a collections of other subevents, for example, a conference event consists of multiple presentations. In brief, '*event*' might have different definitions and semantics in different application domains and contexts.

Despite the fact that there are various definitions and meanings for the term '*event*', in this thesis, we simply take this definition from WordNet[1]: *"something that happens at a given place and time"*. This definition is also commonly used in the Information Retrieval community [35], and especially in Topic Detection and Tracking [3].

From such a definition, an event, or more precisely, a *spatio-temporal event*, is typically described with three types of information: *context*, *time*, and *location*. They are considered three core components of an event description to answer the three following questions about the corresponding fact in the real world: *"What happened?"*, *"When did it happen?"*, and *"Where did it happen?"*.

For example, Figure 3.1 depicts an event about a performance of the band Scorpions in Stuttgart, described using an RDF-based event description. One can see different related information about the events represented as properties, such as the title, the category, or the venue. Such properties can be classified as either contextual, temporal, or spatial information, such as the grouping shown in Figure 3.1.

In the following sections, we will discuss from where to obtain such event data, and how to utilize the data for the purpose of mining interesting event patterns.

3.1.2 Event Data Sources

Nowadays, there is a huge number of data sources available on the Internet that provide event data where each record consist of information about the context, time

[1]Princeton University "About WordNet." WordNet. Princeton University. 2010. http://wordnet.princeton.edu

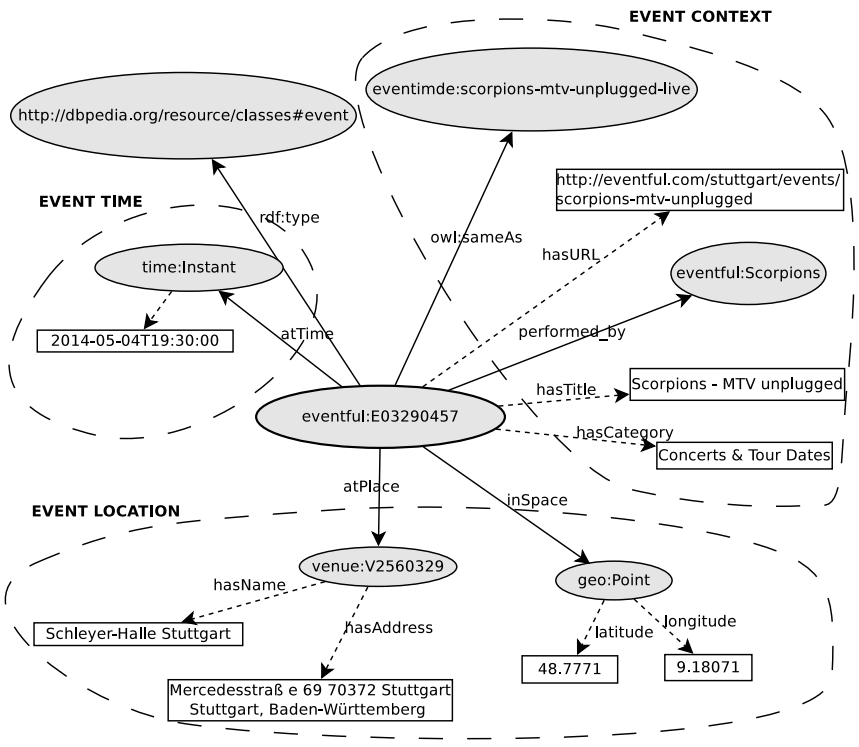


Figure 3.1: Example of an event described using an RDF-based event description.

and location of an event. A media event such as a concert, sports competition, or festival, describing a public activity at a certain place and time, is reported by several websites such as *eventful.com* or *eventbrite.com*. Historical events like wars, battles, or conflicts, where each event consists of an event description, a time interval, and a location, can be found at Linked Open Data sources such as DBPedia [7] or YAGO2 [49]. Data about music performances including titles, descriptions, ticket prices, venues, scheduled time, etc., can be accessed from several websites such as *last.fm* or *eventim.de*. The above data sources are either available for download or they provide flexible APIs for developers to access.

Since there are heterogeneous event data sources, as mentioned above, it is necessary to transform such sources into event data in a uniform format for the purpose of data mining and knowledge discovery. Figure 3.2 shows an overview of our framework for event data processing, where raw event data can be from either a traditional (relational) database or an RDF-based data source using some event ontology such as LODE [101], Event Ontology[2], or the event class of DBpedia [7]. Such an RDF-based data source is obtained from the Linked Open Data cloud using SPARQL endpoints,

[2]http://purl.org/NET/c4dm/event.owl

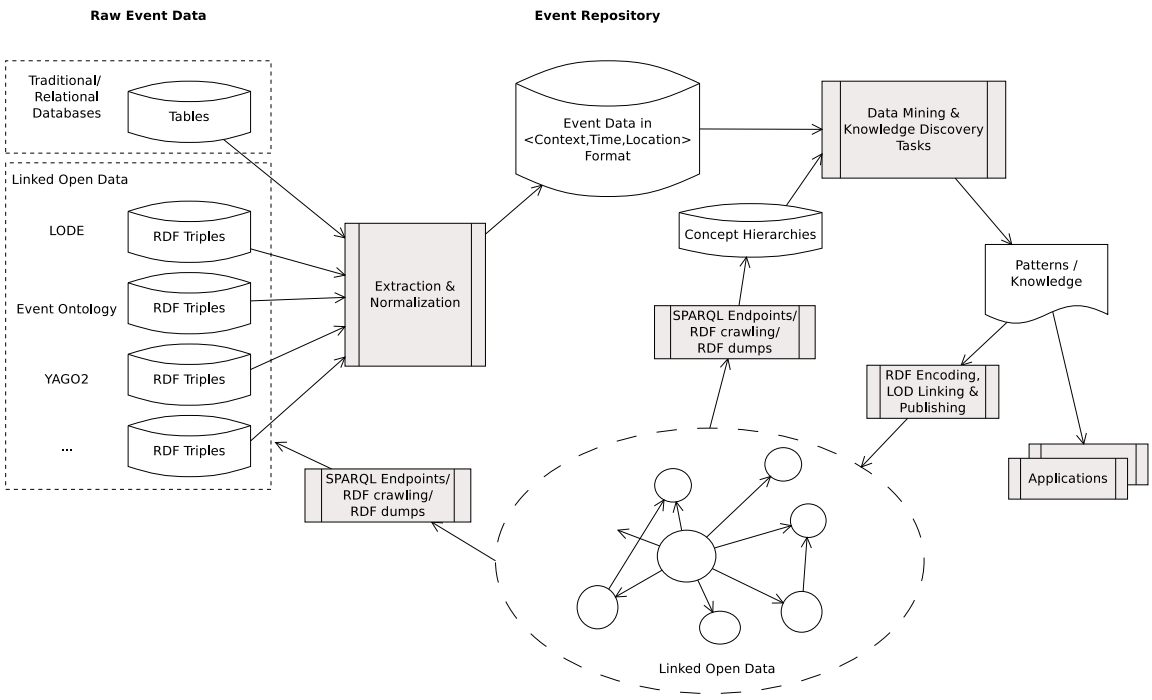


Figure 3.2: Overview of event data processing for data mining and knowledge discovery tasks.

an RDF crawler, or RDF dumps. This process can be considered a preprocessing step including extraction and normalization of raw event data.

The extracted event information together with concept hierarchies, which are also obtained from the Linked Open Data cloud, are input to the data mining and knowledge discovery tasks. These tasks will be the focus of Chapters 4 to 6. The results or outputs of these tasks, i.e., event patterns, knowledge, or semantic annotations, are utilized further in different applications such as recommendation services or business intelligence. In addition, the results can be encoded in RDF, linked to other LOD sources, and published on the Linked Open Data cloud, as shown in Figure 3.2.

In the following section, we build a comprehensive and flexible framework to model events. This framework provides the basis to formulate event patterns which will be discussed in the later chapters.

3.1.3 Event Modeling

As mentioned before, an event description consists of three core components: context, time, and location. To be more specific, an event is described as a tuple $\langle e\_id,\ C,\ T,\ L\rangle$, where e_id is an event identifier, and C, T, and L are the con-

text, time, and location components, respectively. The last three components are described as follows.

- **Context**: Typically, a context is described as a tuple of *concepts*. Here a *concept* is considered a unit of thoughts, expressed as a linguistic term, e.g., '*Metallica*', or '*Heavy metal*'. A context is the component containing information to answer the '*What-question*' about an event. For example, YAGO2 [48] describes an event as a quintuple ⟨Subject, Predicate, Object, Time, Location⟩, where the first three components, i.e., ⟨Subject, Predicate, Object⟩, can be considered a context of an event, for instance, ⟨*'Madly-in-Anger-with-the-World Tour'*, '*performed_by*', '*Metallica*'⟩. Generally, the context component is the basis to specify conceptual relationships among events.

- **Time**: This component is typically described as a *time point* or *time interval* to answer the '*When-question*' about an event. The time component is fundamental to specify temporal relationships among events.

- **Location**: This component contains information to answer the '*Where-question*' about an event. Basically, spatial relationships among events are specified on the basic of their location components.

In the following sections, we describe the above concepts in more detail.

3.2 Concepts, Hierarchies, and Event Contexts

A context of an event description typically contains one or more *concepts*. In this section, we first specify the term '*concept*' used in this thesis. We then formulate hierarchical relationships among *concepts* to build the basis for conceptual relationships among events.

3.2.1 Concepts

There are various points of view to understand the term '*concept*', e.g., as a mental representation, as an ability, or as an abstract object [80]. For the purpose of data mining and knowledge discovery, we simply consider a *concept* a linguistic term denoting a class of things in the real world. For example, the concept '*scientist*' denotes a class of persons who are scientists (e.g., Albert Einstein, Louis Pasteur, Otto Hahn). The concept '*rock concert*' refers to a class of musical performances in the genres of

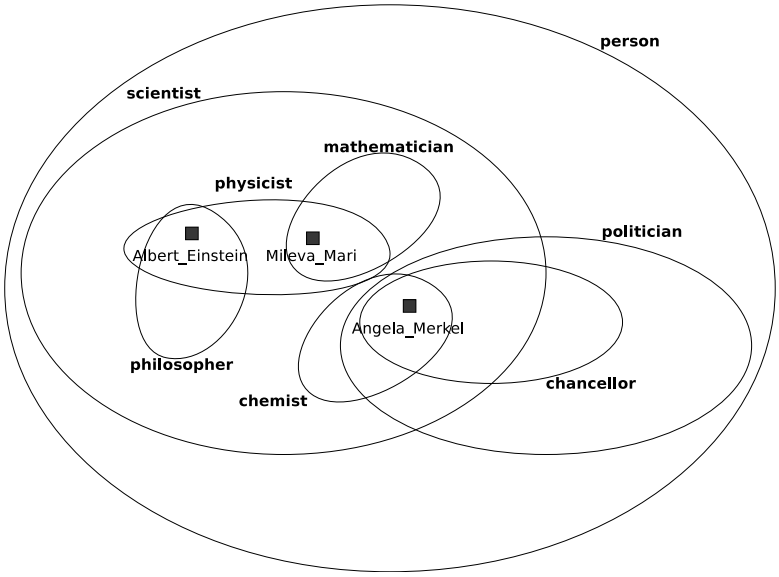


Figure 3.3: An example using a Venn diagram to illustrate entity-category and category-category relationships among 3 entities (*Albert_Einstein*, *Mileva_Mari*, and *Angela_Merkel*) and 8 categories (*person*, *scientist*, *physicist*,...). The entities are represented as elements (squares), and the categories are represented as sets (circles).

Rock 'n' Roll music. Such a class (or concept) is called a *category*, and an instance (e.g., an individual person, a particular music performance, etc.) is called an *entity*.

Typically, a category might contain one or more entities, and an entity might also belong to one or more categories. For instance, the entity '*Albert_Einstein*' belongs to the both categories '*physicist*' and '*philosopher*'. In addition, the *belongs-to* relation can also be applied to categories. For example, both categories '*physicist*' and '*philosopher*' belong to the category '*scientist*'.

To illustrate such *belongs-to* (or *is-a*) relationships, we can use *sets* and *elements* like in the following example. Figure 3.3 shows a Venn diagram, where a set represents a category (e.g., '*scientist*', '*person*', or '*politician*'), and an element represents an entity (e.g., '*Albert_Einstein*', '*Mileva_Mari*', or '*Angela_Merkel*'). For example, the set '*scientist*' is {'*Albert_Einstein*', '*Mileva_Mari*', '*Angela_Merkel*'}; the set '*physicist*' is {'*Albert_Einstein*', '*Mileva_Mari*'}. The fact that an entity belongs to a category is described using a membership relation ($\in$), e.g., '*Albert_Einstein*' $\in$ '*scientist*'. And the fact that a category belongs to another category is described using a set inclusion ($\subset$), e.g., '*scientist*' $\subset$ '*person*'.

Note that the set inclusion ($\subset$) can be applied for entity-category relationships as well if we describe an entity as a unit set, e.g., {'*Albert_Einstein*'} $\subset$ '*scientist*' instead of '*Albert_Einstein*' $\in$ '*scientist*'.

Based on that idea, an entity is also considered a special concept, whose instance is the entity itself. This allows one to uniformly formulate hierarchical relationships between either two categories or a category and an entity. For this, the notations and operators defined for hierarchical relationships among concepts can be applied to both category-category and entity-category relationships. This will be shown in the next section.

In summary, a concept here refers to either an entity or a category; an entity refers to an individual instance; and a category refers to a class of instances.

3.2.2 Concept Hierarchies

As mentioned before, a concept might belong to one or more other concepts. All these *belongs-to relationships* among concepts constitute a *concept hierarchy*. In the literature, there are several ways to specify a concept hierarchy, from a simple solution such as using a taxonomy tree [98, 103] to a more formal, complex one such as using a poset (partially ordered set) [26, 69]. Here we focus on concept hierarchies extracted from Linked Open Data (LOD) sources, where hierarchical relationships among concepts are defined using predicates such as '*is-a*', '*type*', or '*sub-class-of*'. Such predicates can be naturally modelled as edges of a directed graph. In addition, a concept hierarchy from LOD sources typically contains no circular relationships among concepts. Therefore, we simply take directed acyclic graphs (DAGs) to model concept hierarchies. The formal definition is as follows.

Definition 3.1 *(**Concept Hierarchy**) A concept hierarchy $\mathcal{H}$ is a **directed acyclic graph** (V, E), where:*

- *V is a set of concepts, and*
- *an edge $e \in E$ from a node $c_1 \in V$ to another node $c_2 \in V$ represents a **direct generalization relationship** between the concept c_1 and the concept c_2, denoted $c_1 \vdash c_2$.*

Figure 3.4 shows a simple concept hierarchy regarding persons, represented as a DAG, for the concepts in Figure 3.3. In this graph, a node represents a concept (either an entity as a square or a category as a circle), and an edge represents a direct generalization relationship between two concepts, for instance, '*Albert_Einstein*' $\vdash$ '*physicist*', or '*scientist*' $\vdash$ '*person*'. From such edges, a set of direct ancestors of a concept can be computed. For example, direct ancestors of the concept '*Albert_Einstein*' are '*philosopher*' and '*physicist*', based on the hierarchy in Figure 3.4.

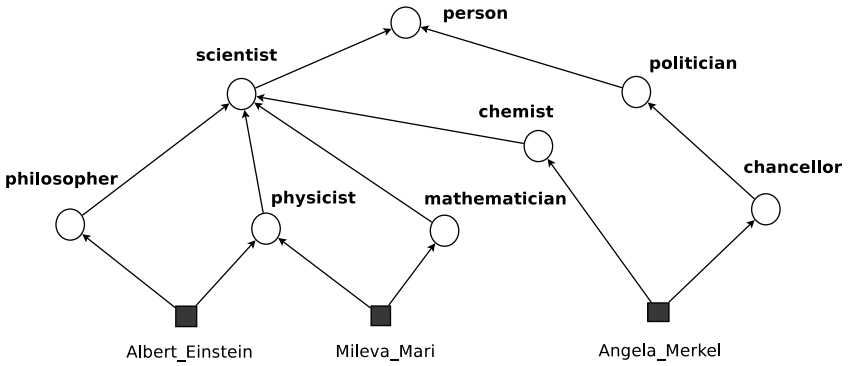


Figure 3.4: Example of a concept hierarchy. Squares represent entities (persons), circles represent categories.

Besides, an *indirect* generalization relationship between two concepts (e.g., '*Albert_Einstein*' and '*person*') can be derived from the hierarchy. We give a formal definition for direct and indirect generalization relationships as follows.

Definition 3.2 *(**Ancestors**) Let $\mathcal{H}$ be a concept hierarchy, and $c \in \mathcal{H}$ be a concept.*

- *A concept $c' \in \mathcal{H}$ is called a **direct ancestor** of the concept c iff $c \vdash c'$.*
- *A concept $c' \in \mathcal{H}$ is called an **ancestor** (direct or indirect) of the concept c, denoted $c \Vdash c'$, iff either c' is a direct ancestor of c or there exist concepts $a_1, a_2, ..., a_n$ ($a_i \in \mathcal{H}$, $1 \leq i \leq n$) such that $c \vdash a_1 \vdash ... \vdash a_n \vdash c'$.*

In other words, there is an indirect relationship from a concept to another concept if and only if there exists a directed path from the first one to the second one. For example, because of the path '*Albert_Einstein*' $\vdash$ '*physicist*' $\vdash$ '*scientist*' $\vdash$ '*person*', we say that the concept '*person*' is an indirect ancestor of the concept '*Albert_Einstein*' (or in other words, the concept '*Albert_Einstein*' belongs to the concept '*person*'). Note that there might exists two or more paths to determine the indirect generalization relationships between two concepts. For example, there are two paths between '*Albert_Einstein*' and '*person*' in the hierarchy depicted in Figure 3.4.

As mentioned before, concepts in an event context specify topics of the corresponding event. To generate event topics later on, we define two *generalization operators* ($\uparrow$ and $\Uparrow$) for computing the set of ancestors of a concept.

Definition 3.3 *(**Concept Generalization Operators**) Let $\mathcal{H}$ be a concept hierarchy, and $c \in \mathcal{H}$ be a concept. The two sets $c^{\uparrow}$ and $c^{\Uparrow}$ consisting of ancestors of the concept c are defined as*

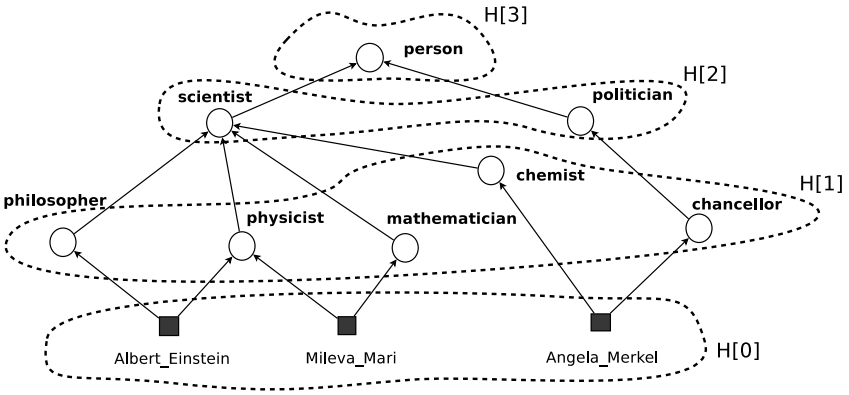


Figure 3.5: Example of concept levels in a hierarchy.

- $c^{\uparrow} := \{c' \in \mathcal{H} \mid c \vdash c'\}$,
- $c^{\Uparrow} := \{c' \in \mathcal{H} \mid c \Vdash c'\}$.

In brief, the set $c^{\uparrow}$ consists of all direct ancestors of the concept c, whereas the set $c^{\Uparrow}$ consists of both direct and indirect ancestors of the concept c. For example, based on the concept hierarchy shown in Figure 3.4, we have '*Albert_Einstein*'$^{\uparrow}$ = {'*philosopher*', '*physicist*'}, and '*Albert_Einstein*'$^{\Uparrow}$ = {'*philosopher*', '*physicist*', '*scientist*', '*person*'}.

In the pattern mining approaches that will be discussed in this thesis, entities are typically generalized to their ancestors to obtain higher levels of abstraction. Since concept hierarchies might be large, considering all ancestors in a pattern mining approach is very time-consuming. On the other hand, the user might be interested in only some specific concept levels. Thus, the levels for concepts need to be specified for the purpose of setting conceptual constraints, i.e., how far a concept can be generalized. They are recursively defined as follows.

Definition 3.4 *(**Concept Level**) Let $\mathcal{H}$ be a concept hierarchy $\mathcal{H}$. The levels 0, 1, 2,... of $\mathcal{H}$, denoted $\mathcal{H}[0], \mathcal{H}[1], \mathcal{H}[2], ...$, respectively, are the sets of concepts recursively defined as*

- $\mathcal{H}[0] := \{c \in \mathcal{H} \mid \nexists c' \in \mathcal{H} : c' \vdash c\}$,
- $\mathcal{H}[i] := \{c \in \mathcal{H} \mid \exists c' \in \mathcal{H}[i-1] : c' \vdash c\}$ *($i \geq 1$).*

*Given a concept $c \in \mathcal{H}[i]$ ($i \geq 0$), c is called at the **level** i of the hierarchy $\mathcal{H}$. The largest value i such that $\mathcal{H}[i] \neq \{\}$ is called the **top level** of $\mathcal{H}$.*

Generally speaking, the level 0 of a concept hierarchy consists of all entities, and the levels $i \geq 1$ consist of categories. Figure 3.5 shows the concept levels for the hierarchy in Figure 3.4, where the set $\mathcal{H}[0]$ including all entities is computed first; the set $\mathcal{H}[1]$ is computed from the set of $\mathcal{H}[0]$; and so on. Note that in this example, the set of $\mathcal{H}[i]$ where $i \geq 4$ is empty, and $i = 3$ is the top level of the concept hierarchy.

Summing up, Section 3.2.1 and Section 3.2.2 describe a framework of concepts and concept hierarchies, designed for the purpose of pattern mining from event data. This framework consists of fundamental components to formulate event contexts, described next.

3.2.3 Event Contexts

As shown in Section 3.1.3, the event context is one of the three core components describing an event. Based on the framework of concepts and concept hierarchies in the previous sections, we consider a context as a thematic component in an event description. We also define generalization operators ($\uparrow$ and $\Uparrow$) to specify hierarchical relationships among event contexts.

An event description might contain many properties, for instance, an event title, a full description, a topic, participants, etc. For a particular application domain, the user might be interested in only some properties describing the context of an event. Thus, we model an event context as a tuple of event properties of interest, where each component of the tuple is taken from an individual concept hierarchy. A formal definition of event contexts is given as follows.

Definition 3.5 *(**Event Context**) Given n concept hierarchies $\mathcal{H}_1$, $\mathcal{H}_2$,...,$\mathcal{H}_n$, an **event context** is described as a tuple $\langle c_1, c_2,...,c_n\rangle$, where $c_i \in \mathcal{H}_i$ $(1 \leq i \leq n)$.*

*A **context framework** $\mathcal{CF}$ is a set of event contexts, defined as*

$$\mathcal{CF} := \{\langle c_1, c_2, ..., c_n\rangle \mid \forall i \in \{1, 2, ..., n\},\ c_i \in \mathcal{H}_i\}.$$

*A subset of $\mathcal{CF}$, called the **most specialized event contexts (MSECs)**, consists of event contexts whose concepts are entities, defined as*

$$\mathcal{CF}[0] := \{\langle c_1, c_2, ..., c_n\rangle \mid \forall i \in \{1, 2, ..., n\},\ c_i \in \mathcal{H}_i[0]\}.$$

Generally, we use event contexts to describe event topics. Specially, the most specialized ones (MSECs) are used to formulate event instances, which are crucial to define instances of patterns later on.

Based on Definition 3.5, we now describe hierarchical relationships among event contexts. Similar to concepts, we first define two generalization operators ($\uparrow$ and $\Uparrow$) for contexts.

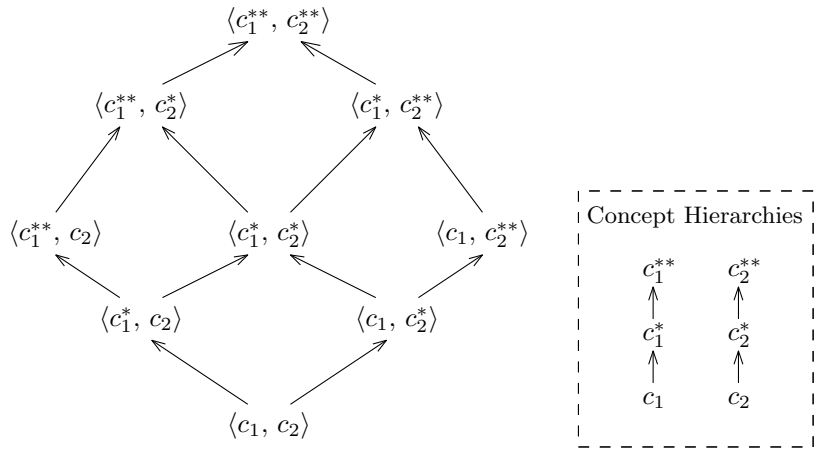


Figure 3.6: Example of a context lattice generated from a context $\langle c_1,\ c_2\rangle$.

Definition 3.6 ***(Context Generalizations)*** *Given n concept hierarchies $\mathcal{H}_1$, $\mathcal{H}_2$,...,$\mathcal{H}_n$, and an event context $C = \langle c_1,\ c_2,...,c_n\rangle$, where $c_i \in \mathcal{H}_i$ $(1 \leq i \leq n)$, the two sets $C^{\uparrow}$ and $C^{\Uparrow}$ consists of event contexts, defined as*

- $C^{\uparrow} := \bigcup_{i=1}^{n} F_i$, *where* $F_i = \{\langle c_1,\ c_2,...,c_i',...,c_n\rangle \mid \forall c_i' \in \mathcal{H}_i : c_i \vdash c_i'\}$,
- $C^{\Uparrow} := \{\langle c_1^*,\ c_2^*,...,c_n^*\rangle \mid \forall i \in \{1, 2, ..., n\},\ c_i^* = c_i\ \vee\ c_i \Vdash c_i^*\} \setminus \{\langle c_1,\ c_2,...,c_n\rangle\}$.

In Definition 3.6, the set $C^{\uparrow}$ consists of contexts obtained from the context C by replacing only one component by its direct ancestor (i.e., c_i by c_i') whereas the set $C^{\Uparrow}$ consists of contexts obtained from the context C by replacing one or more components by their ancestors. Obviously, the set $C^{\uparrow}$ is a subset of the set $C^{\Uparrow}$.

Based on the operators $\uparrow$ and $\Uparrow$ for contexts, generalized/specialized relationships among contexts can be described as follows. Given two event contexts C_1 and C_2, we say that C_1 is more specialized than C_2 (or C_2 is more generalized than C_1) if and only if $C_2 \in C_1^{\Uparrow}$. In that case, the context C_2 is called a *generalization* of the context C_1. Specially, the context C_2 is called a *direct generalization* of the context C_1 if $C_2 \in C_1^{\uparrow}$. Following this, event contexts can be generated step-by-step from the specialized ones to the more generalized ones.

To illustrate how to generate generalizations step-by-step for a given context, we use two simple hierarchies for concepts in Figure 3.6. This figure shows a lattice generated from a context in the form of a pair of two components $\langle c_1,\ c_2\rangle$, where the concepts c_1 and c_2 can be generalized as $c_1 \vdash c_1^* \vdash c_1^{**}$ and $c_2 \vdash c_2^* \vdash c_2^{**}$, respectively. In this example, $\langle c_1,\ c_2\rangle^{\uparrow}$ is a set consisting of two elements, i.e., $\{\langle c_1^*,\ c_2\rangle, \langle c_1,\ c_2^*\rangle\}$. Next, the $\uparrow$ operator is again applied for the contexts in that set to generate the next direct generalizations, i.e., $\{\langle c_1^{**},\ c_2\rangle, \langle c_1^*,\ c_2^*\rangle, \langle c_1,\ c_2^{**}\rangle\}$, and so on. Finally,

the generated lattice of contexts contains all generalization of the original context $\langle c_1,\ c_2 \rangle$, as shown in Figure 3.6.

In some applications, patterns related to specific concepts (e.g., '*scientist*') are more interesting and meaningful than other patterns related to general concepts (e.g., '*person*'). However, the latter patterns dominate in mining results that employ a statistical interestingness measure like the support [65]. Therefore, the step-by-step context generation method mentioned in the previous paragraph is crucial to develop an algorithm that eliminates redundancies and produces only patterns containing the most specialized event contexts.

3.3 Location and Time

Location and Time are two other core components of an event description. In the following, we introduce respective frameworks to represent locations and time of events.

3.3.1 Spatial Framework for Event Location

This section describes a framework to model locations of events in the presence of multiple granularities.

Basically, locations can be specified at different levels of granularity. A *spatial granularity* such as country, state, or city represents a *partition* of the space (spatial domain) in regions, called *spatial entities*. We give a formal definition of spatial granularities, and based on that, we define locations as follows.

Definition 3.7 (Spatial Granularity) *A spatial granularity $\mathcal{G}_\mathcal{S}$ is a finite set of* ***disjoint regions****, called* ***spatial entities****, of the space.*

For example, if we define a spatial granularity '*Country*' as a set of countries and '*City*' as a set of cities in the world, then '*Germany', 'France*', or '*USA*' are spatial entities of the granularity '*Country*', whereas '*Heidelberg', 'Paris*', or '*Las Vegas*' are spatial entities of the granularity '*City*'. Generally, a spatial entity is identified in combination with a granularity. For example, '*Berlin*' can be considered a state or a city of Germany. For this, a location is defined as a pair of a spatial granularity and a spatial entity. A formal definition is as follows.

Definition 3.8 (Location) *Given a spatial granularity $\mathcal{G}_S$, a location is a pair $\langle \mathcal{G}_\mathcal{S}, g \rangle$, where g is a spatial entity in $\mathcal{G}_\mathcal{S}$.*

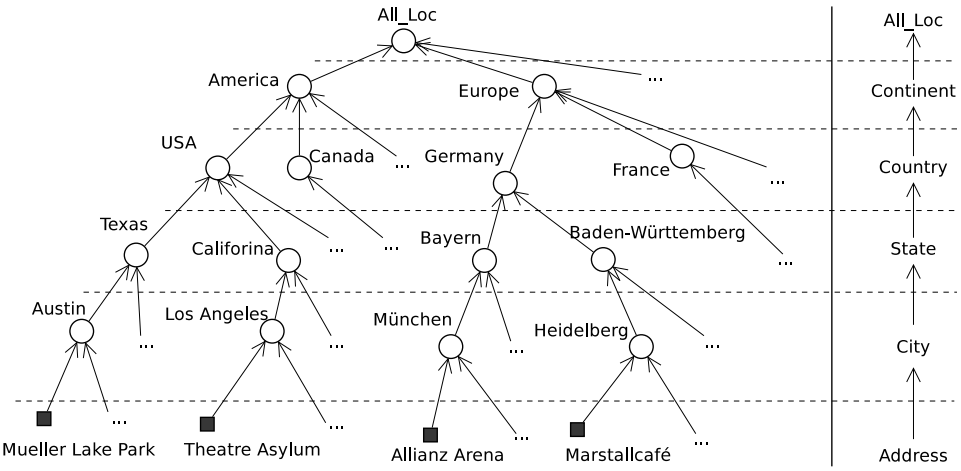


Figure 3.7: Example of a location hierarchy. A representation based on relationships among spatial entities is on the left, where locations of the finest granularity are marked with squares, and other locations of the coarser granularity are marked with circles. On the right, a schema-based representation is shown for the hierarchy.

In a location representation, the spatial entity g, typically represented as an identifier, can be used to lookup the *spatial extent* for the corresponding region. Spatial relationships between two locations are then determined based on their spatial extents. The mappings from the set of identifiers to the set of spatial extents might be explicitly provided in knowledge base datasets or can obtained using geo-tagger tools such as Google Places[3].

For convenience, in our examples from now on, we use strings (semantic names) to represent spatial entities in location representations when no ambiguity of such names exists, for instance, $\langle$'*Country*', '*Germany*'$\rangle$ or $\langle$'*City*', '*Heidelberg*'$\rangle$.

A *spatial framework*, denoted $\mathcal{SF}$, is a set of locations, formally defined as follows.

Definition 3.9 (Spatial Framework) *Given a set of spatial granularities $\mathcal{GS}$, a* ***spatial framework*** *is a set of locations $\mathcal{SF} = \{l_1, l_2, ...\}$, where each location $l_i = \langle G, g\rangle$ is a pair of a granularity $G \in \mathcal{GS}$ and a spatial entity $g \in G$.*

Similar to concepts, locations can be organized in hierarchies, called spatial hierarchies. Figure 3.7 shows an example of a spatial hierarchy consisting of hierarchical relationships among spatial entities. Here, each spatial entity of the granularity '*Address*' is mapped to a spatial entity of the granularity '*City*'; each spatial entity of the granularity '*City*' is mapped to a spatial entity of the granularity '*State*'; and so on.

[3]https://developers.google.com/places

On the right, Figure 3.7 also depicts another representation of the hierarchy, which is based on hierarchical relationships among spatial granularities (e.g., '*Address*', '*City*', '*Country*'). Such a representation is called *a schema hierarchy*.

Different from concept hierarchies, a spatial hierarchy is typically provided by the user for data mining and knowledge discovery tasks in the form of a schema hierarchy, instead of a DAG of spatial entities. However, hierarchical relationships among spatial entities, e.g. '*Heidelberg*' $\rightarrow$ '*Baden-Württemberg*' $\rightarrow$ '*Germany*', need to be known in mining algorithms. The question is how to derive hierarchical relationships among spatial entities from a schema hierarchy.

To solve the above problem, we first specify relationships among spatial granularities. Based on that, we then define spatial hierarchies. Finally, we describe how to infer relationships among spatial entities on the basis of containment relationships (see the 9-intersection model [32]) among spatial extents.

Given two spatial granularities (e.g., '*City*' and '*Country*'), a common question is, which one is finer (or coarser). For this, we define *finer-than relationships* among spatial granularities as follows.

Definition 3.10 (Spatial Finer-than Relationship) *Given two spatial granularities $\mathcal{G}_1$ and $\mathcal{G}_2$ of the same space, $\mathcal{G}_1$ is **finer than** $\mathcal{G}_2$ (or $\mathcal{G}_2$ is **coarser than** $\mathcal{G}_1$), denoted $\mathcal{G}_1 \sqsubset \mathcal{G}_2$, iff for each region $r_1 \in \mathcal{G}_1$, there exists a region $r_2 \in \mathcal{G}_2$ such that r_1 is covered by r_2.*

Definition 3.10 specifies a finer-than relationship between two spatial granularities based on containment relationships among spatial entities. For example, since each city is contained in some country, the granularity '*City*' is finer than the granularity '*Country*'. Note that there is at most one spatial entity of the coarser granularity covering a given spatial entity of the finer granularity, since spatial entities of the same granularity do not overlap.

Considering such finer-than relationships allows one to define a valid schema hierarchy. For example, '*Address*' $\rightarrow$ '*City*' $\rightarrow$ '*Country*' is a valid hierarchy, whereas '*City*' $\rightarrow$ '*Address*' is not. A formal definition of valid schema hierarchies for locations, or in short, spatial hierarchies, is given as follows.

Definition 3.11 (Spatial Hierarchy) *Let $\mathcal{GS}$ be a set of spatial granularities of the same space. A **spatial hierarchy** is defined as a directed acyclic graph where a node is a spatial granularity in $\mathcal{GS}$, and an edge between two spatial granularities $\mathcal{G}_1$ and $\mathcal{G}_2$ ($\mathcal{G}_1 \sqsubset \mathcal{G}_2$) represents a direct generalization relationship, denoted $\mathcal{G}_1 \vdash \mathcal{G}_2$.*

If there is a path from $\mathcal{G}_1$ to $\mathcal{G}_2$ in the hierarchy $\mathcal{GS}$, then we say that they have a ***generalization relationship****, denoted $\mathcal{G}_1 \Vdash \mathcal{G}_2$.*

Typically, there is a node in a DAG of a spatial hierarchy such that other nodes have a path to it. That node is called '*All_Loc*', and it represents for the coarsest granularity of the hierarchy.

Based on a spatial hierarchy, locations of a granularity can be mapped to ones of coarser granularities. Such mappings are based on two generalization operators ($\uparrow$ and $\Uparrow$) formally defined as follows.

Definition 3.12 *(**Location Generalizations**) Let $\mathcal{H}_\mathcal{S}$ be a spatial hierarchy, and $l = \langle \mathcal{G}, g \rangle$ be a location of the granularity $\mathcal{G} \in \mathcal{H}_\mathcal{S}$. The two sets $l^\uparrow$ and $l^\Uparrow$ consisting of locations are defined as*

- $l^\uparrow = \langle \mathcal{G}, g \rangle^\uparrow := \{ \langle \mathcal{G}', g' \rangle \mid \mathcal{G} \vdash \mathcal{G}' \wedge g' \in \mathcal{G}' \ \wedge \ g \text{ is covered by } g' \}$,
- $l^\Uparrow = \langle \mathcal{G}, g \rangle^\Uparrow := \{ \langle \mathcal{G}', g' \rangle \mid \mathcal{G} \Vdash \mathcal{G}' \wedge \ g' \in \mathcal{G}' \ \wedge \ g \text{ is covered by } g' \}$.

For example, based on the hierarchy in Figure 3.7, we have

$\langle$'*City*', '*Heidelberg*'$\rangle^\uparrow = \{\langle$'*State*', '*Baden-Württemberg*'$\rangle\}$, and

$\langle$'*City*', '*Heidelberg*'$\rangle^\Uparrow = \{ \langle$'*State*', '*Baden-Württemberg*'$\rangle$, $\langle$'*Country*', '*Germany*' $\rangle$, $\langle$'*Continent*', '*Europe*'$\rangle$, $\langle$'*All_Loc*', '*'$\rangle\}$,

$\langle$'*All_Loc*', '*'$\rangle^\uparrow = \{\}$.

We now have a comprehensive framework to model locations of events. Similarly, we formally define *time* and *a temporal framework* to model event time in the next section.

3.3.2 Temporal Framework for Event Time

Similar to locations, a time instance can be specified at different granularities such as day, month, or year. To formalize temporal granularities, we start with the concept of *chronons*, as follows.

We model the time line as a set of non-decomposable units of time, called *chronons*, with a total order $\leq_t$. Note that the meaning of the term '*chronon*' here is different from in other sciences such as Quantum Physics. Here, chronon is the smallest unit of time that can be represented in a particular application, for instance, second, day, or month.

Definition 3.13 (Chronons) *A **time domain** is a finite set $\mathcal{T}$ of non-decomposable units of time, called **chronons**, with a total order $\leq_t$.*

Since the set of chronons $\mathcal{T}$ is totally ordered under $\leq_t$, the following statements hold for all chronons a, b, and c in $\mathcal{T}$:

- If $a \leq_t b$ and $b \leq_t a$ then $a =_t b$ (antisymmetry);
- If $a \leq_t b$ and $b \leq_t c$ then $a \leq_t c$ (transitivity);
- $a \leq_t b$ or $b \leq_t a$ (totality).

The relation $\leq_t$ among chronons is important to define temporal relationships between time instances such as the Allen's relationships [4] later on.

Based on the concept of chronons, we now define temporal granularities.

Definition 3.14 (Temporal Granularity) *Given a time domain $\mathcal{T}$, a temporal granularity $\mathcal{G}_\mathcal{T}$ of $\mathcal{T}$ is a finite set of **disjoint subsets** of $\mathcal{T}$, each subset is called **time instances**.*

For example, if a day is chosen as a chronon, and a time domain $\mathcal{T}$ is the set of all days from 2000 to 2013, then the granularities $\mathcal{G}_{Month}$ and $\mathcal{G}_{Year}$ are two sets:

$$\begin{aligned}\mathcal{G}_{Month} = \{\{&\textbf{2000-01}\text{-}01, \textbf{2000-01}\text{-}02, ..., \textbf{2000-01}\text{-}31\},\\ &\{\textbf{2000-02}\text{-}01, \textbf{2000-02}\text{-}02, ..., \textbf{2000-02}\text{-}31\}, ...,\\ &\{\textbf{2013-12}\text{-}01, \textbf{2013-12}\text{-}02, ..., \textbf{2013-12}\text{-}31\}\}\end{aligned}$$

$$\begin{aligned}\mathcal{G}_{Year} = \{\{&\textbf{2000}\text{-}01\text{-}01, \textbf{2000}\text{-}01\text{-}02, ..., \textbf{2000}\text{-}12\text{-}31\},\\ &\{\textbf{2001}\text{-}01\text{-}01, \textbf{2001}\text{-}01\text{-}02, ..., \textbf{2001}\text{-}12\text{-}31\}, ...,\\ &\{\textbf{2013}\text{-}01\text{-}01, \textbf{2013}\text{-}01\text{-}02, ..., \textbf{2013}\text{-}12\text{-}31\}\}.\end{aligned}$$

That is, the set $\mathcal{G}_{Month}$ consists of 168 (12*14) elements (an element represents a time instance), where each element is a set of days in a month (e.g., January 2000, February 2000, or December 2013). The set $\mathcal{G}_{Year}$ consists of 14 elements, where each element is a set of days in a year (e.g., 2000, 2001, or 2013).

Typically, an event occurrence can be described as *a time point* or *a time interval*. Thus, we first formalize time points on the basis of the definition of temporal granularities, and we then define time intervals. A time point is modeled as a time instance of some granularity, formally defined as follows.

Definition 3.15 (Time Point) *A **time point** is a pair $\langle \mathcal{G}_\mathcal{T}, I \rangle$, where $\mathcal{G}_\mathcal{T}$ is a temporal granularity and I is a time instance in $\mathcal{G}_\mathcal{T}$.*

For example, $\langle \mathcal{G}_{Month}$, 'January 2012'$\rangle$ and $\langle \mathcal{G}_{Year}$, '2012'$\rangle$ are two time points of the granularity $\mathcal{G}_{Month}$ and $\mathcal{G}_{Year}$, respectively. From now on, we use names for temporal granularities in our examples, e.g., *Day* for $\mathcal{G}_{Day}$, or *Month* for $\mathcal{G}_{Month}$.

Based on time points, we now define time intervals. For this, we first specify order relationships among time points as follows.

Given two time points $t_1 = \langle \mathcal{G}_1, I_1 \rangle$ and $t_2 = \langle \mathcal{G}_2, I_2 \rangle$, we say that t_1 is less than t_2 ($t_1 <_t t_2$) if and only if each chronon in I_1 has a $<_t$ relationship with each chronon in I_2. A time interval is specified based on two time points t_s and t_e (called start-point and end-point, respectively) such that $t_s \leq_t t_e$. We give a formal definition as follows.

Definition 3.16 (Time Interval) *A **time interval** of a temporal granularity $\mathcal{G}_\mathcal{T}$ is a tuple $\langle \mathcal{G}_\mathcal{T}, I_s, I_e \rangle$ where I_s and I_e are time instances of $\mathcal{G}_\mathcal{T}$, and $\langle \mathcal{G}_\mathcal{T}, I_s \rangle \leq_t \langle \mathcal{G}_\mathcal{T}, I_e \rangle$. The time points $\langle \mathcal{G}_\mathcal{T}, I_s \rangle$ and $\langle \mathcal{G}_\mathcal{T}, I_e \rangle$ are called **start-time** and **end-time**, respectively.*

For example, the time interval $\langle$ *Year*, 2011, 2013$\rangle$ represents a time interval of three years, from 2011 to 2013.

Based on a particular application domain, time components of events can be specified using either time points or time intervals. For this, we define two different temporal frameworks, one for time points and another one for time intervals.

Definition 3.17 (Time Point Framework) *Given a set of temporal granularities $\mathcal{GT}$, a **time point framework** $\mathcal{TF}_P$ is a set of time points $\langle \mathcal{G}, I \rangle$, where $\mathcal{G} \in \mathcal{GT}$ and $I \in \mathcal{G}$.*

Definition 3.18 (Time Interval Framework) *Given a set of temporal granularities $\mathcal{GT}$, a **time interval framework** $\mathcal{TF}_I$ is a set of time intervals $\langle \mathcal{G}, I_s, I_e \rangle$, where $\mathcal{G} \in \mathcal{GT}$, $I_s \in \mathcal{G}$, $I_e \in \mathcal{G}$, and $I_s <_t I_e$.*

Similar to spatial granularities, temporal granularities can be organized in a hierarchy as well. To define such a hierarchy, we specify a finer-than relationship among temporal granularities on the basis of inclusion of chronon sets as follows.

Definition 3.19 (Temporal Finer-than Relationship) *Given two temporal granularities $\mathcal{G}_1$ and $\mathcal{G}_2$ of the same time domain, $\mathcal{G}_1$ is **finer than** $\mathcal{G}_2$ (or $\mathcal{G}_2$ is **coarser than** $\mathcal{G}_1$), denoted $\mathcal{G}_1 \sqsubset \mathcal{G}_2$, iff for each time instances $I_1 \in \mathcal{G}_1$, there exists a time instance $I_2 \in \mathcal{G}_2$ such that $I_1 \subset I_2$.*

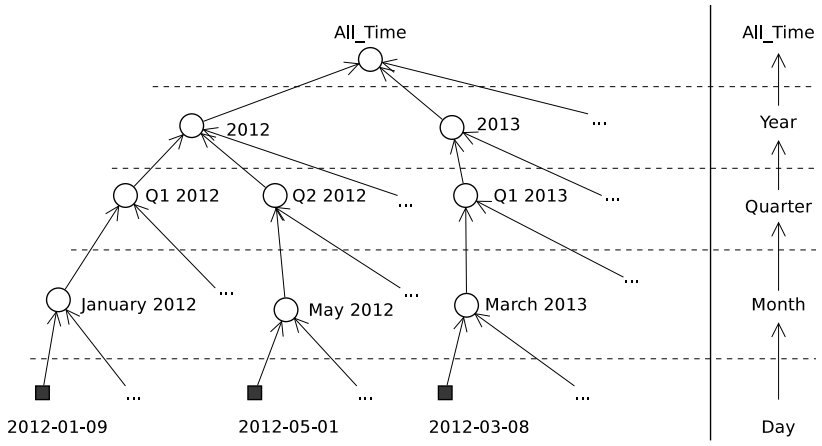


Figure 3.8: Example of a temporal hierarchy.

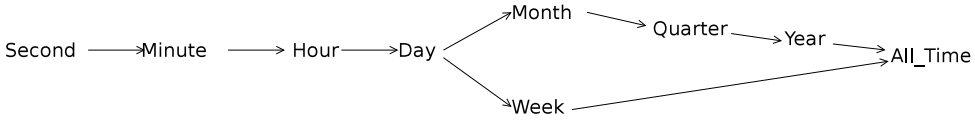


Figure 3.9: Example of a temporal hierarchy represented as a DAG.

For example, the granularity *Month* is finer than the granularity *Year*, and it is coarser than the granularity *Day*. Such finer-than relationships allow one to define a temporal hierarchy, where the lowest level is the finest granularity and the highest level of the hierarchy is the coarsest one. Typically, the highest level of a temporal hierarchy is called *All_Time*. On the right, Figure 3.8 depicts a temporal hierarchy, where the finest granularity is *Day* and the coarsest one is *All_Time*. On the left, one can see each time instance at a lower level of the hierarchy is mapped to one time instance at the higher one.

Similar to spatial hierarchies, a temporal hierarchy can be a DAG, for example, the one shown in Figure 3.9. A formal definition is given as follows.

Definition 3.20 (Temporal Hierarchy) *Let $\mathcal{GT}$ be a set of temporal granularities of the same time domain. A* ***temporal hierarchy*** *is defined as a directed acyclic graph where a node is a temporal granularity in $\mathcal{GT}$, and an edge between two temporal granularities $\mathcal{G}_1$ and $\mathcal{G}_2$ ($\mathcal{G}_1 \sqsubset \mathcal{G}_2$) represents a direct generalization relationship, denoted $\mathcal{G}_1 \vdash \mathcal{G}_2$.*

If there is a path from $\mathcal{G}_1$ to $\mathcal{G}_2$ in the hierarchy $\mathcal{GT}$, then we say that they have a ***generalization relationship****, denoted $\mathcal{G}_1 \Vdash \mathcal{G}_2$.*

For example, Figure 3.9 shows some generalization relationships such as $Hour \vdash Day$, $Day \vdash Month$, and $Day \Vdash Year$, whereas $Hour \nvdash Month$, $Week \nVdash Month$, and $Week \nVdash Year$.

Based on a temporal hierarchy, a time point (or a time interval) can be generalized to coarser granularities. Similar to locations, we also use the generalization operators $\uparrow$ and $\Uparrow$ to convert a time point or a time interval into coarser granularities. They are formally defined as follows.

Definition 3.21 *(**Time Point Generalizations**) Let $\mathcal{H}_\mathcal{T}$ be a temporal hierarchy, and $t = \langle \mathcal{G}, I \rangle$ be a time point of the granularity $\mathcal{G} \in \mathcal{H}_\mathcal{T}$. The two sets $t^\uparrow$ and $t^\Uparrow$ consisting of time points are defined as*

- $t^\uparrow = \langle \mathcal{G}, I \rangle^\uparrow := \{ \langle \mathcal{G}', I' \rangle \mid \mathcal{G}' \vdash \mathcal{G} \ \wedge \ I' \in \mathcal{G}' \ \wedge \ I' \supset I \ \}$,
- $t^\Uparrow = \langle \mathcal{G}, I \rangle^\Uparrow := \{ \langle \mathcal{G}', I' \rangle \mid \mathcal{G}' \Vdash \mathcal{G} \ \wedge \ I' \in \mathcal{G}' \ \wedge \ I' \supset I \ \}$.

Definition 3.22 *(**Time Interval Generalizations**) Let $\mathcal{H}_\mathcal{T}$ be a temporal hierarchy, and $t = \langle \mathcal{G}, I_s, I_e \rangle$ be a time interval of the granularity $\mathcal{G} \in \mathcal{H}_\mathcal{T}$. The two sets $t^\uparrow$ and $t^\Uparrow$ consisting of time intervals are defined as*

- $t^\uparrow = \langle \mathcal{G}, I_s, I_e \rangle^\uparrow := \{ \langle \mathcal{G}', I'_s, I'_e \rangle \mid \mathcal{G}' \vdash \mathcal{G} \ \wedge \ I'_s, I'_e \in \mathcal{G}' \ \wedge \ I'_s \supset I_s \ \wedge \ I'_e \supset I_e \ \}$,
- $t^\Uparrow = \langle \mathcal{G}, I_s, I_e \rangle^\Uparrow := \{ \langle \mathcal{G}', I'_s, I'_e \rangle \mid \mathcal{G}' \Vdash \mathcal{G} \ \wedge \ I'_s, I'_e \in \mathcal{G}' \ \wedge \ I'_s \supset I_s \ \wedge \ I'_e \supset I_e \ \}$.

For example, using the hierarchy in Figure 3.8, we have

$\langle Day, \text{`}2012\text{-}01\text{-}09\text{'} \rangle^\uparrow = \{ \langle Month, \text{`}January\ 2012\text{'} \rangle \}$,

$\langle Day, \text{`}2012\text{-}01\text{-}09\text{'} \rangle^\Uparrow = \{ \langle Month, \text{`}January\ 2012\text{'} \rangle, \langle Quarter, \text{`}Q1\ 2012\text{'} \rangle, \langle Year, \text{`}2012\text{'} \rangle, \langle All\_Time, \text{`}*\text{'} \rangle \}$, and

$\langle Day, \text{`}2012\text{-}01\text{-}09\text{'}, \text{`}2012\text{-}01\text{-}20\text{'} \rangle^\uparrow = \{ \langle Month, \text{`}January\ 2012\text{'}, \text{`}January\ 2012\text{'} \rangle \}$.

3.4 Events and Event Templates

Based on the frameworks for concepts, time, and locations described in the previous sections, we now formulate events and then introduce the notion of event templates to describe topics of events. Both concepts of events and event templates build the basis for our pattern mining approaches in the later chapters.

3.4.1 Events

As mentioned before, an event description includes three core components: a context, a time instance, and a location. Based on the framework for concepts, time, and locations, we formally define events as follows.

Definition 3.23 (Event) *Let $\mathcal{CF}$, $\mathcal{SF}$, and $\mathcal{TF}$ be a context framework, a spatial framework, and a temporal framework, respectively.*

*An **event** is specified as a tuple $\langle e\_id, C, T, L\rangle$, where e_id is an identifier to distinguish an event from another, $C \in \mathcal{CF}$ is a context, $T \in \mathcal{TF}$ is a time point (or a time interval), and $L \in \mathcal{LF}$ is a location.*

For example, the event describing the marriage between Albert Einstein and Mileva Mari is described as the tuple $\langle$#00347, $\langle$*'Albert_Einstein', 'isMarriedTo', 'Mileva_Mari'*$\rangle$, $\langle$*Year*, 1903, 1920$\rangle$, $\langle$*Country*, *'Switzerland'*$\rangle\rangle$, where $\langle$*'Albert_Einstein', 'isMarriedTo', 'Mileva_Mari'*$\rangle$ is the context, $\langle$*Year*, 1903, 1920$\rangle$ is the time interval, and $\langle$*Country*, *'Switzerland'*$\rangle$ is the location of the event.

Based on the definition of events, we now define *event datasets*, which are input for pattern mining approaches, as follows.

Definition 3.24 (Event Dataset) *Let $\mathcal{CF}$, $\mathcal{SF}$, and $\mathcal{TF}$ be a context framework, a spatial framework, and a temporal framework, respectively.*

*An **event dataset**, denoted $\mathcal{D}_E$, is a set of events, where each event is a tuple $\langle e\_id, C, T, L\rangle$, where $C \in \mathcal{CF}$, $T \in \mathcal{TF}$, and $L \in \mathcal{LF}$.*

Generally, all events in the same event dataset are described by using the same context framework, the same spatial framework, and the same temporal framework. The context, time, and location components of these events can be generalized to higher level of abstractions based on the operators $\Uparrow, \uparrow$ mentioned in earlier sections. This feature is the basis of formulating *event templates* to describe event topics in the next section.

3.4.2 Event Templates

Events as introduced above can be considered base facts in a knowledge base, and they build the basis for our pattern mining approaches. Components of such patterns are obtained by generalizing contexts, time, and locations by their direct or indirect generalizations to obtain so-called *event templates*.

Definition 3.25 *(**Event Template**) Let $\mathcal{CF}$, $\mathcal{SF}$, and $\mathcal{TF}$ be a context framework, a spatial framework, and a temporal framework, respectively.*

*An **event template (ET)** is specified as a triple $\langle C, T, L\rangle$, where $C \in \mathcal{CF}$ is a context, $T \in \mathcal{TF}$ is a time point (or a time interval), and $L \in \mathcal{SF}$ is a location.*

Obviously, in comparison to events, ETs, whose context, time, and location can be taken at any level of abstraction on the basis of given hierarchies, do not explicitly exist in a dataset but are to be derived from events. An event might "produce" many different ETs and an ET might be derived from different events. The following definition makes this aspect more precise, utilizing the generalization operators introduced earlier.

Definition 3.26 *(**ET Instance**) Given an event template $f = \langle C^*, T^*, L^*\rangle$, the ET f is **supported** by an event $e = \langle id, C, T, L\rangle$ (or e is an **instance** of f), denoted $e \Vdash f$, iff $(C^* \in C^{\Uparrow} \cup \{C\}) \wedge (T^* \in T^{\Uparrow} \cup \{T\}) \wedge (L^* \in L^{\Uparrow} \cup \{L\})$.*

To avoid duplicates in deriving event templates as parts of patterns later in our approaches, we assume a total order $\prec$ on the set of ETs. Without loss of generality, we assume a total order on each $\mathcal{CF}$, $\mathcal{SF}$, and $\mathcal{TF}$. The total order $\prec$ on ETs is then defined as the lexicographical order on the elements of the Cartesian product $\mathcal{CF} \times \mathcal{SF} \times \mathcal{TF}$.

As in an event template, the elements of the components can be taken from any levels of the underlying hierarchies, it is reasonable to introduce a (generalization) relationship between ETs. Given an ET f, the set $f^{\uparrow}$ consists of all ETs obtained from f by replacing one component, i.e., context, time, or location, by its direct generalizations based on the underlying hierarchies. The set $f^{\Uparrow}$ then simply is the closure of f.

Definition 3.27 *(**ET Generalizations**) Given an ET $f = \langle C,\ T,\ L\rangle$ specified based on a given context framework $\mathcal{CF}$, a temporal framework $\mathcal{TF}$, and a spatial framework $\mathcal{SF}$, the **generalizations** of f, denoted $f^{\uparrow}$ and $f^{\Uparrow}$, are defined as:*

- $f^{\uparrow} := \{\langle C', T, L\rangle \mid C' \in C^{\uparrow}\} \cup \{\langle C, T', L\rangle \mid T' \in T^{\uparrow}\} \cup \{\langle C, T, L'\rangle \mid L' \in L^{\uparrow}\}$.
- $f^{\Uparrow} := \{\langle C^*, T^*, L^*\rangle \mid C^* \in C^{\Uparrow} \cup \{C\},\ T^* \in T^{\Uparrow} \cup \{T\},\ L^* \in L^{\Uparrow} \cup \{L\}\} \setminus \{\langle C, T, L\rangle\}$.

To illustrate how to obtain event templates from an event, we use an example shown in Figure 3.10, where the first ET (i.e., $f = \langle C,\ T,\ L\rangle$) is obtained by combining the components C, T, and L of a given event $e = \langle id,\ C,\ T,\ L\rangle$. Next, the

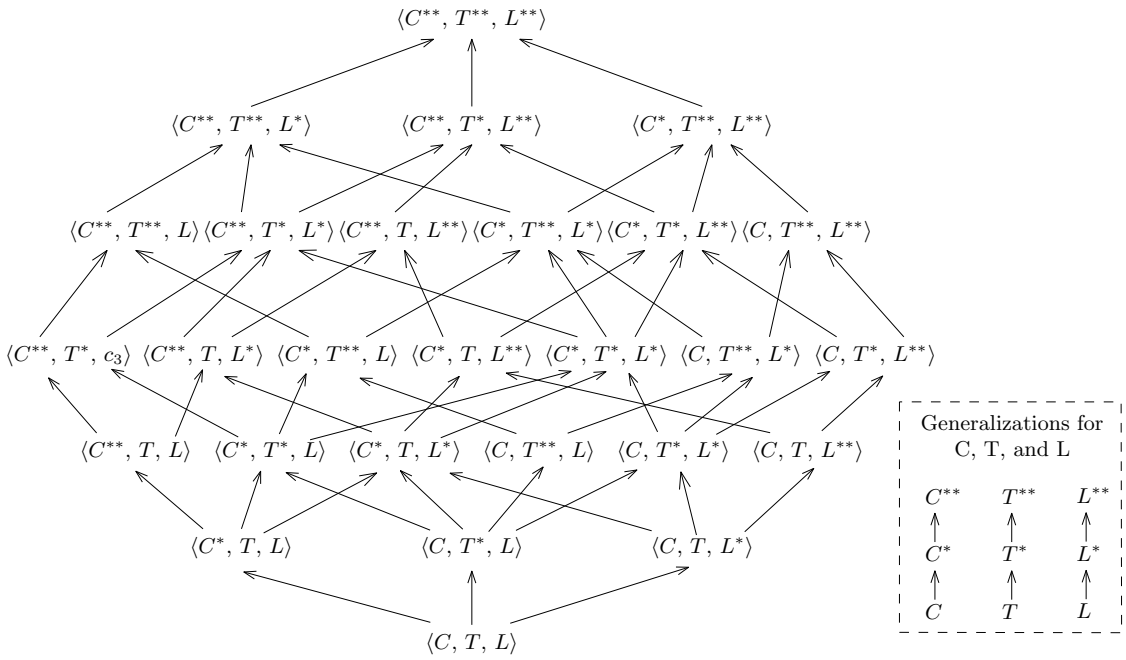


Figure 3.10: Example of an ET lattice generated from an event $\langle id,\ C,\ T,\ L\rangle$. Generalizations of C, T, and L are obtained based on the sub-figures on the right.

generalization $\uparrow$ is applied on the ET f to generate the next generations of ETs. This process repeats until the topmost generalizations of C, T and L (i.e., C^{**}, T^{**} and L^{**}) are considered. We finally obtain a lattice of ETs consisting of all generalizations of the first ETs. Note that the event e is an instance of any ET in the lattice.

After introducing the notions of events and event templates as basis for our pattern mining approaches, we now focus on relationships among events that are determined by context, time, and location components of events. Such relationships are important to model instances of event patterns in the pattern mining approaches that we focus on in this thesis.

3.5 Event Relationships

As mentioned before, events are modeled with three core components (Context, Time, and Location). Given two events, considering relationships between their locations allows one to determine spatial relationships between the two events (e.g., the proximity of locations using the Euclidean distance) on the basis of the corresponding spatial extents. Similarly, their contexts and their time instances help to determine conceptual and temporal relationships between events. Suitably combining the three kinds of relationships allows one to determine how related two events are for a specific

application. Moreover, based on such relationships, different kinds of constraints can be specified for the mining approaches to obtain interesting results with respect to some user requirements. We describe spatial, temporal, and conceptual relationships between events in the following.

3.5.1 Spatial Relationships

The spatial framework described in Section 3.3.1 supports mapping a location to a *spatial extent*. For a particular application, spatial extents can be one of basic spatial data types such as *points*, *lines*, or *regions* [40]. For example, the location Berlin (Germany) can be considered a point in the Euclidean plane or a region with a boundary. Specifying relationships between spatial extents is fundamental to determine how spatially related the corresponding events are.

Representing spatial objects and formulating spatial relationships among them are well studied in existing work [32, 40, 119]. In such approaches, spatial predicates describing relationships among spatial objects such as INSIDE, ADJEACENT, or INTERSECTS are formally defined. Utilizing these predicates allows one to flexibly deal with various kinds of relationships among locations of events. Since this chapter aims at building a general framework for an event model, we use an abstract relation $\mathcal{R}_s$ to represent a spatial relationship for event locations. Such a relationship can be one of the following basic classes [40]:

- Distance-based relationships: for example, if the Euclidean distance of two point-based locations is no less than a given threshold (e.g., 1 km), then the locations has a $\mathcal{R}_s$-relationship.
- Topological relationships: the relation $\mathcal{R}_s$ is specified by some predicates such as INSIDE, OVERLAP, or DISJOINT, and these predicates are given on the basis of a specific application domain.
- Directional relationships: similar to topological relationships, the relation $\mathcal{R}_s$ is specified by some given predicates such as ABOVE, BELOW, or SOUTH_OF.

Based on the notation of a relation $\mathcal{R}_s$ for locations, we give a formal definition of spatial relationships between two events.

Definition 3.28 (Spatial Relationship) *Given two events* $\langle e_1, C_1, T_1, L_1\rangle$ *and* $\langle e_2, C_2, T_2, L_2\rangle$*, the events* e_1 *and* e_2 *have a* ***spatial relationship****, denoted* $(e_1, e_2) \in \mathcal{R}_s$*, iff the locations* L_1 *and* L_2 *have an* $\mathcal{R}_s$*-relationship.*

Using Definition 3.28, one can determine spatial relationships among events by employing a given relation $\mathcal{R}_s$ for spatial extents. These relationships are crucial to define instances of patterns later on.

3.5.2 Temporal Relationships

Section 3.3.2 describes two temporal frameworks to formulate event time, one for time points and the other for time intervals. Thus, we here introduce temporal relationships among events by employing the relationships between either two time points or two time intervals, described as follows.

In a time point framework, a time point is specified as a pair of a temporal granularity (e.g., *Month*) and a time instance (e.g., June 2013). We formulate a relation $\leq_t$ for time points by using relationships between chronons that are described in Section 3.3.2.

Definition 3.29 (Time Point Orders) *Given two time points* $P_1 = \langle \mathcal{G}, I_1 \rangle$ *and* $P_2 = \langle \mathcal{G}, I_2 \rangle$ *of the same granularity* $\mathcal{G}$*, we say that* $P_1 \leq_t P_2$ *iff* $\forall c_1 \in I_1, c_2 \in I_2 : c_1 \leq_t c_2$.

Given two time points $Q_1 = \langle \mathcal{G}_1, I_1 \rangle$ *and* $Q_2 = \langle \mathcal{G}_2, I_2 \rangle$ *of two different granularities, assume* $\mathcal{G}_1 \Vdash \mathcal{G}_2$*, we say that* $Q_1 \leq_t Q_2$ *iff* $Q'_1 \leq_t Q_2$*, where* $Q'_1 = \langle \mathcal{G}_2, I'_1 \rangle$ *is the mapping of* Q_1 *to the granularity* $\mathcal{G}_2$.

The relation $\leq_t$ can be applied for both cases, i.e., two time points of the same granularity and two time points of different granularities, for example,

$\langle Month, \text{`June 2013'} \rangle <_t \langle Month, \text{`July 2013'} \rangle$,

$\langle Month, \text{`June 2013'} \rangle <_t \langle Year, \text{`2014'} \rangle$, and

$\langle Month, \text{`June 2013'} \rangle =_t \langle Year, \text{`2013'} \rangle$.

Using the relation $\leq_t$ for time points, we extend the basic predicates for time points and time intervals [4] to support multiple granularities, as given in Table 3.1. These predicates are utilized further to define a temporal relationship $\mathcal{R}_t$ for events later on.

Similar to distance-based relationships for locations, a temporal relationship $\mathcal{R}_t$ for events can be defined by employing a "distance" function between two time points. Such a function will be used to specify the length of time windows in which two events are to be related, for example, two events are temporally related if the duration

Predicate	**Meaning**
For time points P_1 and P_2	
EQUAL(P_1, P_2)	$P_1 =_t P_2$
BEFORE(P_1, P_2) AFTER(P_2, P_1)	$P_1 <_t P_2$
For time intervals I_1 and I_2	
EQUAL(I_1, I_2)	start-time$(I_1) =_t$ start-time(I_2) $\wedge$ end-time$(I_1) =_t$ end-time(I_2)
BEFORE(I_1, I_2) AFTER(I_2, I_1)	end-time$(I_1) <_t$ start-time(I_2)
MEET(I_1, I_2) MET_BY(I_2, I_1)	end-time$(I_1) =_t$ start-time(I_2)
OVERLAPS(I_1, I_2) OVERLAPPED_BY(I_2, I_1)	start-time$(I_1) <_t$ start-time(I_2) $\wedge$ start-time$(I_2) <_t$ end-time(I_1) $\wedge$ end-time$(I_1) <_t$ end-time(I_2)
STARTS(I_1, I_2) STARTED_BY(I_2, I_1)	start-time$(I_1) =_t$ start-time(I_2) $\wedge$ end-time$(I_1) <_t$ end-time(I_2)
DURING(I_1, I_2) CONTAINS(I_2, I_1)	start-time$(I_2) <_t$ start-time(I_1) $\wedge$ end-time$(I_1) <_t$ end-time(I_2)
FINISHES(I_1, I_2) FINISHED_BY(I_2, I_1)	start-time$(I_2) <_t$ start-time(I_1) $\wedge$ end-time$(I_1) =_t$ end-time(I_2)

Table 3.1: Temporal predicates for time points and time intervals. The predicates for time intervals are formulated on the basis of the Allen's temporal relations [4].

between their occurrence time is not more than 3 days. To handle such conditions, we define a distance function, called *t-dist*.

Definition 3.30 *(**Time Point Distances**) Given two time points $P_1 = \langle \mathcal{G}, I_1 \rangle$ and $P_2 = \langle \mathcal{G}, I_2 \rangle$ of the same granularity $\mathcal{G}$ (assume $P_1 \leq_t P_2$), the **distance** between P_1 and P_2, denoted t-dist(P_1, P_2) or t-dist(P_2, P_1), is defined as*

$$\text{t-dist}(P_1, P_2) = \text{t-dist}(P_2, P_1) := \mid \{Q = \langle \mathcal{G}, I \rangle \mid P_1 \leq_t Q <_t P_2\} \mid.$$

Given two time points $Q_1 = \langle \mathcal{G}_1, I_1 \rangle$ and $Q_2 = \langle \mathcal{G}_2, I_2 \rangle$ of two different granularities, assume $\mathcal{G}_1 \Vdash \mathcal{G}_2$, the distance between Q_1 and Q_2 is defined as

$$\text{t-dist}(Q_1, Q_2) = \text{t-dist}(Q_2, Q_1) := \text{t-dist}(Q_1', Q_2),$$

where $Q_1' = \langle \mathcal{G}_2, I_1' \rangle$ is the mapping of Q_1 to the granularity $\mathcal{G}_2$.

If two given time points are of the same granularity, then the distance between them is calculated by counting the time points between them of that granularity. Because the set of time points of a granularity is finite, this value is either zero or a positive, definite integer. For example, the distance between $\langle Year, \text{'2011'} \rangle$ and $\langle Year, \text{'2012'} \rangle$ is 1 (year).

If two given time points are of different granularities, the time point of the finer granularity will be converted so that both time points have the same granularity before calculating the distance. For example, the distance between $\langle Month, \text{'June 2011'}\rangle$ and $\langle Year, \text{'2012'}\rangle$ is 1 (year) because $\langle Month, \text{'June 2011'}\rangle$ is mapped to $\langle Year, \text{'2011'}\rangle$, and $t\text{-}dist(\langle Year, \text{'2011'}\rangle, \langle Year, \text{'2012'}\rangle) = 1$.

In summary, an $\mathcal{R}_t$ relation for event time can be defined in various ways such as using temporal predicates or a distance function, depending on the particular application domain. Based the $\mathcal{R}_t$ relation, we finally give a formal definition of temporal relationships between two events.

Definition 3.31 (Temporal Relationship) *Given two events $\langle e_1, C_1, T_1, L_1\rangle$ and $\langle e_2, C_2, T_2, L_2\rangle$, the events e_1 and e_2 have a **temporal relationship**, denoted $(e_1, e_2) \in \mathcal{R}_t$, iff the time components T_1 and T_2 have an $\mathcal{R}_t$-relationship.*

3.5.3 Conceptual Relationships

Similar to time and locations, contexts can be utilized to specify relationships among events. The relationships derived from contexts are called *conceptual relationships*, specified on the basis of relationships among concepts in a given hierarchy. In this section, we first describe some measures of *similarity* and *relatedness* for concepts and then define conceptual relationships for contexts of events.

Measuring the semantic similarity and relatedness of concepts is a generic problem for many Natural Language Processing (NLP) tasks [88]. For example, the concept '*heavy metal*' is considered more similar to the concept '*rock*' than to the concept '*jazz*' when employing a music genre hierarchy. Generally, relatedness is more general than similarity since two concepts can be related even they are not similar. For example, the concept '*heavy metal*' is considered related to concepts such as '*drummer*', '*bassist*', or '*music performance*'. A similarity or relatedness measure is a measure that quantifies the degree of similarity or relatedness between two concepts on the basis of the underlying concept hierarchy.

The importance of such measures to many NLP applications has led to many proposals, such as, such as path based approaches [68, 121] or information content based approaches [70, 56, 96]. Since we are focusing on how to specify conceptual relationships among events for the purpose of pattern mining approaches, not aiming at proposing a new measure for concepts, we utilize the state-of-the-art measures in the existing work for concepts to formulate *conceptual relationships for contexts*. The availability of open source software packages such as WordNet::Similarity [88] allows

one to consider a similarity/relatedness measure as an abstract function that outputs a real value for two given concepts. Such a function can be implemented using any state-of-the-art measure based on a particular application. Based on that, we assume a function, called $sim(c_1, c_2) \in [0, 1]$, to compute the similarity/relatedness between two concept c_1 and c_2. Using this function for concepts, we define a similarity measure for event contexts.

Definition 3.32 (Context Similarity) *Given two event contexts* $C = \langle c_1, c_2, ..., c_n \rangle$ *and* $C' = \langle c'_1, c'_2, ..., c'_n \rangle$ *in the same context framework* $\mathcal{CF}$*, the* ***similarity*** *between* C *and* C' *is defined as*

$$c\text{-}sim = \frac{\sum_{i=1}^{n} \alpha_i sim(c_i, c'_i)}{n} \qquad \in [0, 1], \tag{3.1}$$

where $\alpha_1, \alpha_2, ..., \alpha_n \in [0, 1]$ *are weighting factors such that* $\sum_{i=1}^{n} \alpha_i = 1$.

The context similarity between two event contexts is computed by using the *sim* function for concepts, with some weighting factors that are defined for a particular application scenario. For example, assume that contexts are of the form ⟨*Subject, Predicate, Object*⟩, and if we focus only on the *Subject* component, the weighting factors are set to $\alpha_1 = 1$, $\alpha_2 = 0$, and $\alpha_3 = 0$.

Using the c-sim function for contexts, we formally define conceptual relationships for events.

Definition 3.33 (Conceptual Relationship) *Given two events* $\langle e_1, C_1, T_1, L_1 \rangle$ *and* $\langle e_2, C_2, T_2, L_2 \rangle$*, and a threshold* $\theta \in [0, 1]$*, the events* e_1 *and* e_2 *have a* ***conceptual relationship****, denoted* $(e_1, e_2) \in \mathcal{R}_c$*, iff* $c\text{-}sim(C_1, C_2) \geq \theta$.

In summary, to specify conceptual relationships among event contexts, a similarity/relatedness measure for concepts is employed. The flexibility of our framework is that one can select such a measure from various existing approaches that is suitable for a specific application scenario.

3.6 Event and ET Constraints

In this section, we introduce constraints that can be specified by the user in order to guide the search for interesting patterns. These constraints are formulated for either events or event templates.

3.6.1 Spatial, Temporal, and Conceptual Constraints

In practice, users might be interested in patterns that are valid in a specific time interval and/or in a specific geographic region and simply refer to a specific concept. These requirements can be specified as the following constraints on events.

- **Temporal Constraint**: A temporal constraint specifies a *time interval of interest* with a start-time and an end-time, where only events occurring in this interval are considered to mine patterns. To specify this constraint, the user can select either a pair of a start-time and an end-time (e.g., from 2011-01-01 to 2012-12-31) or a start time and a duration (e.g., from 2011-01-01 for 2 years).

- **Spatial Constraint**: The user might be interested in events occurring in a specific geographic region. Such a region of interest can be specified by using a rectangle (e.g., the user draws a bounding box on a map), or simply by giving names of instances (e.g., 'Germany' or 'Munich').

- **Conceptual Constraint**: From the concept hierarchies for event contexts, the user might select some concepts that she is interested in. Then only events related to these concepts, i.e., events having paths to one of the concepts, are considered when mining for respective patterns.

Generally, the user can specify multiple constraints of the above types and combine them using logical connectives. Only events satisfying such conditions are considered further to mine patterns.

3.6.2 Constraints on Event Templates

As discussed before, event templates (ETs), generalized from events by using hierarchies, play an important role in constituting patterns. Based on the constraints described in the previous section, events of interest can be selected to generalize ETs. However, the user might also want to specify how far an event can be generalized. For example, in some case patterns related to concepts that are too general might not be of interest to the user. For this, we describe the following constraints on ETs based on time, location, and context components.

Typically, time and location hierarchies are given in a schema-based representation, such as *Day*→*Month*→*Year* or *City*→*State*→*Country*. Thus, specifying how far time and locations can be generalized is simply specifying level names, e.g., only generalize time components to *Month*, and location components to *State*.

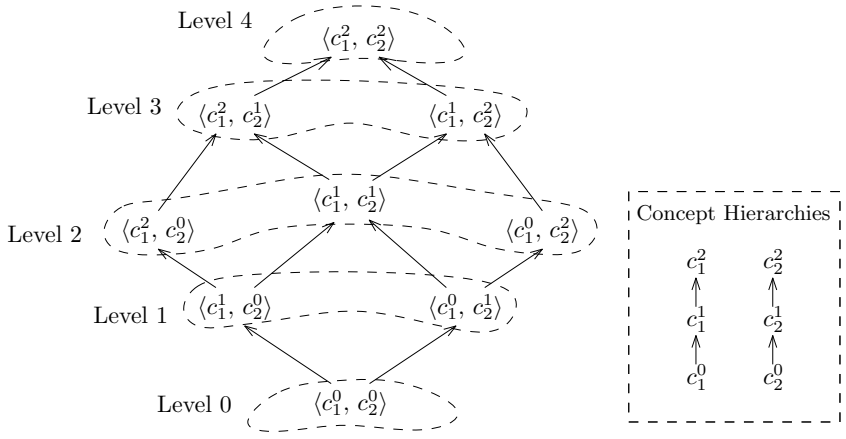


Figure 3.11: Example of contexts generalized from a context $\langle c_1^0, c_2^0\rangle$ at different levels. A superscript indicates the level of the corresponding concept.

Recall that an event context is a tuple of concepts, where each component can be generalized based on a respective concept hierarchy. Different from hierarchies for time and locations, a hierarchy for concepts is given as a DAG, where a node is a concept, and an edge represents an *'is-a'* relationship between two concepts. There is no 'level name' like in schema-based hierarchies for time and locations. However, based on Definition 3.4, one can compute a number (e.g., 0, 1, 2, etc.) to indicate the level of a given concept. Rather than individually specifying how far each component in an event context can be generalized, we first define levels for event contexts, and then specify a maximum level for them.

Generally speaking, the level of a given context is the minimum number of steps to obtain that context from a tuple of respective entities. Here, replacing one component of the tuple by its direct ancestor is considered a step. Recall that the level of an entity is zero (the lowest level), and the level of a category is recursively computed from the levels of its descendants, as described in Definition 3.4. As an example, Figure 3.11 shows a context lattice generated from a context in the form of a pair $\langle c_1^0,\ c_2^0\rangle$, where c_1^0 and c_2^0 are entities that can be generalized as $c_1^0 \vdash c_1^1 \vdash c_1^2$ and $c_2^0 \vdash c_2^1 \vdash c_2^2$, respectively. Note that in this example, a superscript indicates the level of the corresponding concept. One can see that the level of a context is the sum of the levels of the components of that context. For example, the levels of the contexts $\langle c_1^1, c_2^0\rangle$, $\langle c_1^1, c_2^1\rangle$, and $\langle c_1^2, c_2^1\rangle$ are 1, 2, and 3, respectively. We formally define context levels as follows.

Definition 3.34 ***(Context Level)*** *Given a context* $C = \langle c_1, c_2, ..., c_n \rangle$*, the level of* C *is defined as*

$$level(C) := \sum_{i=1}^{n} level(c_i), \tag{3.2}$$

where $level(c_i)$ *is the level of the i-th concept of the context* C *(*$i \in \{1, 2, ..., n\}$*).*

Using a threshold $g_{max} \geq 0$, one can now specify to what extent event contexts can be generalized. For example, one can set $g_{max} = 2$ to specify that only event contexts generalized with no more than two steps are considered to form patterns.

3.7 Discussion

This chapter describes a comprehensive framework for an event model that is fundamental for the pattern mining approaches presented in the following chapters. Describing an event with three separate components, i.e., context, time, and location, allows one to flexibly define how events are related in terms of conceptual, temporal, and spatial relationships. Moreover, the three components of events can be individually generalized to higher levels of abstraction to derive event templates describing topics of events. The notations of events and event templates as well as different relationships among them will be utilized to define pattern languages and interestingness measures for the pattern mining approaches presented in the next chapters. To guide the search for interesting patterns, we also allow the user to specify constraints on events and event templates.

Aiming at building a general, flexible framework, all the concepts, notations and relationships here are very generally defined. However, they can easily be specialized for a particular pattern mining approach, as shown in the following chapters.

Chapter 4

Mining Interval-based Event Sequence Patterns

Sequential pattern mining is a problem frequently addressed by the data mining community, where a pattern is in the form of a sequence representing insights or knowledge gained from temporal or spatio-temporal data. Finding such patterns plays an important role for applications such as user behavior analysis, business intelligence, or event prediction. For example, patterns discovered from data of events, such as concerts, festivals, or sports, are often exploited to provide location based services, to suggest products and services to customers, or to predict the behavior of customers.

Traditional spatio-temporal data for mining sequential patterns are typically obtained through observations and simulations where positions of objects, such as areas, vehicles, or persons, are collected over time. In the past couple of years, however, massive amounts of event data have been increasingly created and shared by users on social media channels. Compared to traditional data, as discussed in Chapter 1, exploiting event data for interesting, useful patterns poses additional challenges for several reasons, for example, the presence of hierarchies associated with event components, and the existence of not only spatio-temporal relationships but also conceptual relationships between events.

In this chapter, we utilize the event model described in Chapter 3 to build a framework in support of the discovery of interesting patterns from datasets of events. Different from traditional approaches to mining spatio-temporal data, we focus on mining sequential patterns at different levels of granularity and abstraction by exploiting conceptual, temporal, and spatial hierarchies, which naturally exist as useful background knowledge for event data. We introduce a pattern specification language and propose an algorithmic approach to efficiently extract complex patterns. We

demonstrate the feasibility and utility of our framework using two different real-world datasets from YAGO2 and the Website eventful.com.

Some initial ideas and results presented in the following appeared in a paper by Le and Gertz [65]. Here we extend these ideas by giving: (1) more elaborate explanations for the notations and definitions, (2) more comprehensive experiments, and (3) discussions in more detail about the runtime and experimental results.

4.1 Introduction

Traditional approaches to the discovery of interesting patterns from spatial and spatio-temporal datasets primarily focus on datasets that have been obtained through observation of objects or simulations, e.g., [43, 73, 81]. With the rise of the Semantic Web and the Linked Open Data (LOD) cloud [46, 10], new data sources come into play. These data sources come in the form of knowledge bases typically consisting of billions of facts. Mining such a large-scale datasets itself presents a challenge (see, e.g., [57]). Additionally, some knowledge bases have recently extended descriptions of facts by spatial and temporal components, such as YAGO2 [48]. Such extensions to facts describe when and where a fact was or will be valid, often resembling the description of an event. Finding interesting spatio-temporal patterns in such event data presents a challenge for several reasons. Some of them originate from mining traditional spatio-temporal datasets, such as the absence of transactions for mining co-locations or establishing suitable (statistical) measures to characterize interesting, useful patterns.

A new challenge, however, is that knowledge bases as mentioned above provide concept hierarchies or category systems, e.g., Wikipedia categories. Similar to the role of object features in, e.g., co-location pattern mining, categories embedded in hierarchies can now be used to specify patterns that include concepts from such hierarchies, thus, allowing to derive patterns at different levels of granularity and abstraction.

For the purpose of mining spatio-temporal patterns from knowledge base data, this chapter specializes the concepts and notations for events and event templates that are generally defined in Chapter 3. Event templates, basically derived from events by using concept, time, and location hierarchies, play an important role for forming complex patterns that are the focus of this work. We present an approach for mining interesting patterns from knowledge base data about events, where the durations of events play an important role in formulating interval-based patterns, for example, in formulating a pattern "*a music concert is performed after a cultural event and both of*

them occur during a festival". For this, we define a class of patterns, called *interval-based event sequence patterns*, where each pattern represents both spatio-temporal proximity and conceptual relatedness of events.

In summary, the contributions of this chapter are as follows:

- We propose a novel approach to the discovery of patterns representing spatio-temporal proximity and conceptual relatedness of events. For this, we specialize the concepts and notations of events and event templates generally defined in Chapter 3.
- We introduce a pattern specification language to describe interval-based relationships among events.
- We propose a suitable interestingness measure for candidate patterns that allows an efficient pruning during the process of generating candidate patterns from events.
- We demonstrate the feasibility and utility of our approach using subsets of the YAGO2 knowledge base and event data from the Website eventful.com.

In the next section, we review related work. In Section 4.3, we detail the notations of events and event templates by employing the concepts and notations introduced in Chapter 3. In Section 4.5, we present our pattern specification language, followed by algorithms to mine such patterns in Section 4.6. After presenting experimental results in Section 4.7, we summarize this chapter in Section 4.8.

4.2 Related Work

Our work is closely related to approaches in co-location pattern mining, sequential pattern mining, and multilevel sequential pattern mining. Chapter 2 already provided a comprehensive survey of existing approaches related to these topics. Here we briefly describe and relate these approaches to our work that is presented in this chapter.

Spatial co-location patterns are subsets of object features where objects exhibiting these features are frequently located nearby in geographic space, see, e.g., [122, 130] for some recent approaches. One of the challenges to apply traditional frequent itemset mining techniques to spatial datasets is that one has to define interestingness measures without having transactions like in transactional datasets. To tackle this problem, Huang et al. [51] proposed a statistical measure, called *participation index*,

to identify prevalent co-location patterns. This measure works well if features have similar frequencies but it fails in datasets where some features are rare [52]. Furthermore, this measure does not consider concept hierarchies according to which object features are organized. In this chapter, we extend this measure to support event datasets where concept hierarchies play an important role.

Whereas spatial co-location patterns are subsets of object features, sequential patterns are sequences where the order of object features in each sequence is important. Several approaches have been proposed to mine sequential patterns, such as SPADE [131], SPAM [8], or PrefixSpan [89]. Since all these approaches assume that datasets consist of sequences of objects, they cannot be applied to spatio-temporal datasets in which object sequences are not given and simply do not make sense.

For the purpose of mining patterns from spatio-temporal datasets, Huang et al. [53] proposed a framework in which both spatial and temporal relationships among objects are considered. A limitation of this approach is that patterns are time point based sequences. Thus, interval based relationships such as CONTAINS, OVERLAPS, or DURING cannot be handled. Moreover, the framework is not flexible enough to extend patterns to interval based sequences consisting of multiple temporal relationships. To handle interval based relationships, Wu and Chen [120] use sequences of endpoints to represent relationships among time intervals. With some modifications to their approach to support different temporal granularities, we define *temporal arrangements* to describe temporal relationships among time intervals, as a part of our pattern language.

The problem of mining multilevel sequential patterns from multidimensional databases, introduced by Plantevit et al. [92], is also related to our work. Different from other approaches in which patterns are sequences of *atomic objects*, their approach can cover sequential patterns in which each object is *a tuple of attributes*. Although their approach can handle concept hierarchies, their patterns are only able to represent time point based relationships. Moreover, they do not consider spatial relationships and, therefore, their approach cannot be applied for spatio-temporal data.

Similar to our work in terms of mining patterns from knowledge base datasets, previous work such as an approach by Jiang and Tan [57] adopts traditional algorithms to mine association rules from RDF data by assuming that RDF documents are transactions containing facts. In the context of spatio-temporal data, defining transactions is not natural due to the continuity of space and time [53]. Furthermore,

some problems arise such as overlapping transactions or loosing spatio-temporal relationships among objects across transactions.

Before presenting our approach, we introduce some basic concepts and notations in the following section.

4.3 Basic Concepts and Notations

In Chapter 3, we presented a comprehensive framework to model events, where notations such as events, event templates, and event relationships are in general described. As an instance of the framework, this section specializes these notations for the purpose of mining spatio-temporal patterns from a dataset of interval-based events.

In the following, we formulate *interval-based events* by employing the event model. We then introduce the notation of *event cliques* that is fundamental to compute instances of patterns later on.

4.3.1 Interval-based Events

An event occurrence might be described with a *start time* and an *end time* to represent its duration. For example, the Second World War is generally considered to have lasted from 1 September 1939 to 2 September 1945. A football match is typically scheduled at a certain time with a duration of about 2 hours. A concert tour of a rock band might last for few months, announced with a specific start date and end date.

An event described with a start time and an end time, called *an interval-based event*, will be the focus of this chapter. Let $\mathcal{CF}$, $\mathcal{SF}$, and $\mathcal{TF}$ be a context framework, a spatial framework, and a time interval framework (Definitions 3.5, 3.9, and 3.18), respectively. An interval-based event is specified as a tuple $\langle e\_id, C, T, L\rangle$, where the components of the tuple are detailed as follows.

- e_id is an identifier to distinguish an event from another in a given dataset.
- C is a context described as an n-tuple of concepts based on the context framework $\mathcal{CF}$.
- T is a time interval in the time interval framework $\mathcal{TF}$, described as a triple $\langle \mathcal{G}_T, I_s, I_e\rangle$, where $\mathcal{G}_T$ is a temporal granularity (e.g., *Year*), I_s and I_e are two time instances in $\mathcal{G}_T$ such that $I_s \leq_t I_e$ (e.g., 1939 and 1945). Two time points

that are the *start-time* and *end-time* of T are denoted as *start-time*$(T) = \langle \mathcal{G}_T, I_s \rangle$ and *end-time*$(T) = \langle \mathcal{G}_T, I_e \rangle$, respectively.

- L is a location in the spatial framework $\mathcal{LF}$, described as a pair $\langle \mathcal{G}_S, g \rangle$, where $\mathcal{G}_S$ is a spatial granularity, and g is a spatial entity. In this chapter, we focus on distance-based relationships for locations, that is, two events are considered spatially related if they are nearby in space. For this, we assume that the spatial proximity of two events can be determine by some distance function, e.g., the Euclidean distance.

The context, time, and location components of an event can be generalized by using the operators computing direct ($\uparrow$) and all ($\Uparrow$) generalizations. Such a generalization of an event is called an event template (ET). Obviously, multiple ETs can be derived from a single event. Generally speaking, an ET is considered an "event type" or an event topic representing a group of events having some common ancestor.

In our approach, the notion of ETs is the basis for formulating event patterns, where each pattern is a combination of several ETs. This will be described in detail in Section 4.5. Since the number of such combinations is often large, one needs an interestingness measure to filter out uninteresting ones. Such a measure is typically defined based on a concept of *pattern instances*, where each instance is simply a combination of events satisfying some temporal, spatial, and conceptual constraints.

Consequently, in the following section, we focus on temporal, spatial, and conceptual constraints for events, and then introduce a notation of *event cliques* to formulate instances of patterns.

4.3.2 Event Cliques

The objective of our proposed approach is to derive combinations of event templates from events (as instances of these templates) such that the events satisfy some "interesting" temporal, spatial, and conceptual properties. Basically, such properties are formulated in terms of conditions on the temporal, spatial, and conceptual relationships of events. In the following, we detail the idea by introducing these relationships between events.

When examining temporal relationships among events, users might want to specify the length of time windows in which two events are to be (temporally) related, similar to concepts employed in mining sequences [22, 79, 104]. To handle such conditions, we define *an interval distance function* between two events, as an extension of *the time point distance function* described in Definition 3.30 (Section 3.5.2). Basically,

a distance between two time intervals is defined as the duration between two time points; one is either the start-time or end-time of a time interval, and the other is also either the start-time or end-time of the other time interval. In particular, choosing the start-time or end-time for each interval to compute the time point distance depends on the *interval relationship* between the time intervals. Such an interval relationship is specified using one of Allen's predicates, given in Table 3.1 (Section 3.5.2). We now give a formal definition of the interval distance between events.

Definition 4.1 *(**Interval Distance**) Let* $e_1 = \langle id_1, C_1, T_1, L_1 \rangle$ *and* $e_2 = \langle id_2, C_2, T_2, L_2 \rangle$ *be two events specified using the same conceptual, temporal, and spatial frameworks. The **interval distance** between* e_1 *and* e_2*, denoted i-dist(e_1,e_2), is determined as follows:*

- *if the relationship between* T_1 *and* T_2 *is BEFORE, then i-dist(e_1,e_2) = start-time(T_2) − end-time(T_1).*
- *if the relationship between* T_1 *and* T_2 *is AFTER, then i-dist(e_1,e_2) = start-time(T_1) − end-time(T_2).*
- *if the relationship between* T_1 *and* T_2 *is any other case (such as OVERLAPS or DURING), then i-dist(e_1,e_2) = 0.*

Note that the '−' operator between two time points in the above definition is specified using the *t-dist* function defined in Definition 3.30 (Section 3.5.2).

Given the length t_w of time-windows as a threshold, two events e_1 and e_2 are said to be related in time, denoted $(e_1, e_2) \in R_t$, if and only if i-dist(e_1,e_2) $\leq t_w$. We call R_t a *temporal relationship*.

Although two events might be in temporal proximity, the events might be geographically far apart. In order to allow for constraints on the spatial proximity of events, *a threshold* for *the minimum spatial neighborhood* between events involved in building a potential pattern needs to be specified. Recall that distance-based relationships for locations of events are assumed. Thus, in our approach, a spatial relationship R_s between two events e_1 and e_2, denoted $(e_1, e_2) \in R_s$, is formulated by a distance function for the locations and a predefined threshold. That is, two events e_1 and e_2 are said to be related in space, denoted $(e_1, e_2) \in R_s$, if and only if the distance between the two event locations is no less than the threshold value.

Similarly, one can define how two events are conceptually related by employing *the context similarity function* (*c-sim*), described in Section 3.5.3. Given a threshold

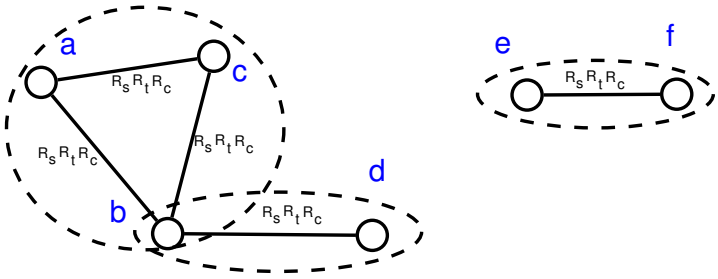


Figure 4.1: Example of maximal event cliques. A node represents an event. An edge between two nodes represents spatial, temporal, and conceptual relationships between the corresponding events satisfying thresholds. Maximal event cliques are enclosed by dashed lines.

for context similarity, two events e_1 and e_2 are said to be conceptually (or topically) related, denoted $(e_1, e_2) \in R_c$, if and only if the context similarity of the two event contexts is no less than the threshold value.

To complete the basis for our pattern definition and mining, we finally combine the above conditions on the temporal proximity, spatial proximity, and conceptual similarity of events forming an instance of a pattern by introducing the notion of an *event clique*.

Definition 4.2 *(**Event Clique**) Let $\mathcal{D}_E$ be an event dataset. Let R_t, R_s, and R_c be a temporal, spatial, and conceptual relationship, respectively.*

*A set of events $E \subset \mathcal{D}_E$, forms an **event clique**, or a **clique** for short, iff $\forall e_1, e_2 \in E$, $e_1 \neq e_2$, the relations $(e_1, e_2) \in R_s$, $(e_1, e_2) \in R_t$, and $(e_1, e_2) \in R_c$ hold.*

Obviously, if a set of events E is an event clique then all subsets of E are also event cliques. Consequently, enumerating all possible cliques from an event dataset often produces redundancies. Thus, a compact set of cliques with no redundancy is more efficient to present the cliques computed from an event dataset. For this, we define the notation of *a maximal event clique*.

Definition 4.3 *(**Maximal Event Clique**) Given an event dataset $\mathcal{D}_E$, an event clique E is called a **maximal event clique**, or a **maximal clique** for short, iff their is no event clique $E' \subset \mathcal{D}_E$ such that $E \subset E'$.*

Figure 4.1 depicts a graph-based representation for a dataset consisting of six events, where a node represent an event, and an edge shows that the two corresponding events have all spatial, temporal, and conceptual relationships (R_s, R_t, R_c). One

can see that the concepts of event cliques and maximal event cliques are similar to the concepts of cliques and maximal cliques in graph theory [12]. Thus, state-of-the-art algorithms for finding maximal cliques in graph, such as the Bron-Kerbosch [14], can be used here.

After introducing the basic concepts and notations that are fundamental to describe event patterns and instances later on, we now focus on how to formulate temporal relationships for a set of interval-based events. For this, we introduce the notation of *temporal arrangements*.

4.4 Temporal Arrangements

In his seminal paper, Allen [4] identifies thirteen possible temporal relations between two time intervals, such as BEFORE, OVERLAPS, or MEETS, which are widely used in temporal reasoning. Recall that in Section 3.5.2, we already extended these relations to support multiple granularities. Basically, in our approach, mapping to the coarser granularity is used to determine the temporal relationship between two time intervals of different granularities. Accordingly, all thirteen temporal relations of Allen can be extended, as shown in Table 3.1.

However, from a semantic point of view, some of them such as MEETS, EQUAL, STARTS, or FINISHES do not make sense in some cases. For example, one cannot say that the two intervals $\langle$*Year*, 2011, 2012$\rangle$ and $\langle$*Month*, December 2011, January 2012$\rangle$ are equal, even though they are identical when the second interval is mapped to the coarser granularity *Year*. Thus, we consider only six types of Allen relations: BEFORE, AFTER, OVERLAPS, OVERLAPPED_BY, DURING, and CONTAINS. The other relations are considered specific cases of these relations. For example, MEETS is considered a specific case of BEFORE, and FINISHES is considered a specific case of DURING, as depicted in Figure 4.2.

Using Allen's relations, the temporal relationship between two events can be determined. However, it is more complicated to express temporal relationships for a set of more than two events because the Allen relations are binary. A naive method using nested combinations of binary relations can be used to express complex relationships among multiple events [59]. For example, complex relationships such as *"event **a** is during event **b**, and both of them are before event **c**"* is represented in an expression ((**a** DURING **b**) BEFORE **c**). However, such an expression might be ambiguous. For example, Figure 4.3 shows two different cases for the same expression ((**a** BEFORE **b**) OVERLAPS **c**).

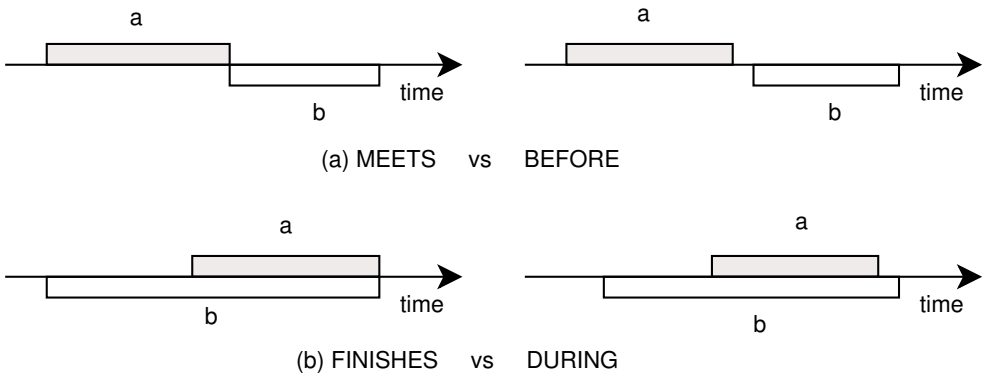


Figure 4.2: MEETS is considered a specific case of BEFORE. FINISHES is considered a specific case of DURING.

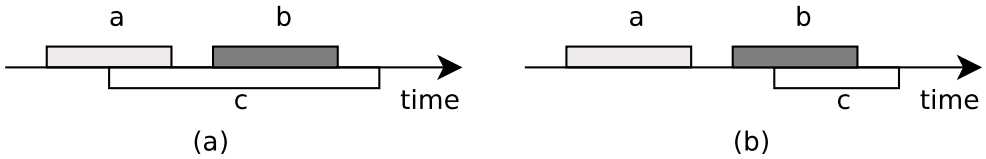


Figure 4.3: Two different temporal relationships can be represented by the same expression ((**a** BEFORE **b**) OVERLAPS **c**).

Although there are various approaches proposed to deal with the ambiguity problem above, such as a matrix of relations [50] or an augmented representation [87], most of them require specialized data structures and algorithms for finding patterns, and as a result, the discovered patterns are also difficult to interpret for the user [82].

Aiming at a simple but unambiguous representation for temporal relationships among a set of events, we introduce the concepts of *start-points* and *end-points*, similar to the work of Wu and Chen [120]. However, based on our framework, we constrain our representation to the description of the six relations mentioned above.

Definition 4.4 *(**Event Start-point and End-point**) An **event-point** of an event e is either a **start-point**, denoted e^+, or an **end-point**, denoted e^-. The timestamps of e^+ and e^-, denoted* time(e^+) *and* time(e^-)*, are the* start-time *and the* end-time *of the time interval of e.*

Given an event-point p (either a start-point or an end-point), the event of p is denoted by event(p)*. The sign of p ('+' if p is a start-point, '−' otherwise) is denoted by* sign(p)*.*

The temporal order of two event-points is basically specified by using the order ($\leq_t$) of their timestamps. However, if their timestamps are equal, other criteria based on the event components of the event-points need to be considered. This is crucial to obtain a unique expression of relationships for a given set of events later on. For this, we assume a syntactical order $\prec_e$ among specifications of events, similar to the syntactical order $\prec$ for event templates introduced in Section 3.4.

Definition 4.5 *(**Event-point Order**) An event-point p_1 has a $\rhd$-*relation *with a event-point p_2, denoted $p_1 \rhd p_2$ (p_1 **followed by** p_2, or p_2 **follows** p_1), iff one of the following conditions is satisfied:*

1. *$time(p_1) <_t time(p_2)$,*
2. *$time(p_1) =_t time(p_2)$ and p_1 is an end-point and p_2 is a start-point,*
3. *$time(p_1) =_t time(p_2)$ and p_1 and p_2 are both start-points (or both end-points) and $event(p_1) \prec_e event(p_2)$.*

Now, using time-point representations, we are able to define an *event-point sequence* describing temporal relationships among events.

Definition 4.6 (Event-point Sequence) *Let E be a set of n events, and Ω be the set of event-points defined as $\Omega = \{e^+ \mid e \in E\} \cup \{e^- \mid e \in E\}$.*

*The **event-point sequence** of E is an arrangement of the $2n$ elements in Ω, denoted $\mathrm{Seq}(E) = p_1 \rhd p_2 \rhd ... \rhd p_{2n}$, with $p_i \in \Omega$, $\forall i \in \{1, ..., 2n\}$, such that the condition $p_i \rhd p_j$ holds for all $1 \leq i < j \leq 2n$.*

Lemma 4.1 *Given a set of events E, the event-point sequence $Seq(E)$ is unique.*

Proof: We prove this lemma by contradiction. Suppose, there are two different event-point sequences $S_1 = p_1 \rhd p_2 \rhd ... \rhd p_{2.|E|}$ and $S_2 = q_1 \rhd q_2 \rhd ... \rhd q_{2.|E|}$. Let i ($1 \leq i \leq 2.|E|$) be the smallest index such that $p_i \neq q_i$, that is, $p_j = q_j (\forall j, 1 \leq j < i)$. Without loss of generality, we assume that $p_i \rhd q_i$. Then, $p_i \rhd q_j$ $\forall j \geq i$, because S_2 is an event-point sequence. Thus, $p_i \notin \{q_i, q_{i+1}, ..., q_{2.|E|}\}$. On the other hand, $p_i \notin \{p_1, p_2, ..., p_{i-1}\} = \{q_1, q_2, ..., q_{i-1}\}$, because S_1 is also an event-point sequence. Therefore, $p_i \notin \{q_1, q_2, ..., q_{i-1}\} \cup \{q_i, q_{i+1}, ..., q_{2.|E|}\} = \Omega$. This contradicts that $p_i \in \Omega$ since p_i is an event-point of an event in E. □

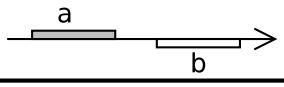
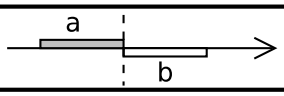
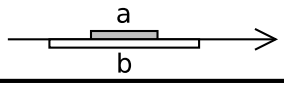
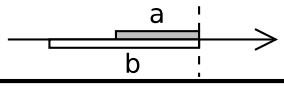
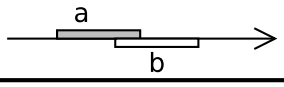
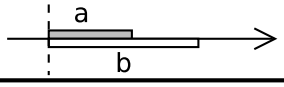
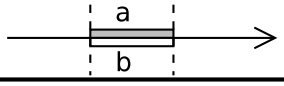
Allen's relationships	Pictorial representations	Event-point Sequences
BEFORE/AFTER		$a^+ \rhd a^- \rhd b^+ \rhd b^-$
MEETS/MET-BY		a BEFORE b b AFTER a
DURING/ CONTAINS		$b^+ \rhd a^+ \rhd a^- \rhd b^-$
FINISHES/ FINISHED-BY		a DURING b b CONTAINS a
OVERLAPS/ OVERLAPPED-BY		$a^+ \rhd b^+ \rhd a^- \rhd b^-$
STARTS/ STARTED-BY		a OVERLAPS b b OVERLAPPED_BY a
EQUAL		

Figure 4.4: Event-point representations for Allen's interval relations. Assume that two events a and b have a syntactical order $a \prec_e b$.

Lemma 4.1 shows that for a given set of events, there is only one expression describing temporal relationships. Following this, the thirteen Allen relations can be reduced to six organized in three pairs; for each pair, only one expression is given, as shown in Figure 4.4.

Moreover, the uniqueness of event-point sequences prevents the ambiguity problem mentioned before. For example, the two cases in Figure 4.3 are presented by two different event-point sequences: (a) $a^+ \rhd c^+ \rhd a^- \rhd b^+ \rhd b^- \rhd c^-$, and (b) $a^+ \rhd a^- \rhd b^+ \rhd c^+ \rhd b^- \rhd c^-$.

Note that for any event e, we always have $e^+ \rhd e^-$.

Similar to the concept of event-points, we also define *ET-points* as ingredients to formulate event patterns.

Definition 4.7 *(**ET Start-point and End-point**) Given an event template f, the **start-point** and **end-point** of the ET f are denoted as f^+ and f^-, respectively. Both f^+ and f^- are called the **ET-points** of the ET f.*

Given an ET-point p (either a start-point or an end-point), the event template of p is denoted by $\mathrm{ET}(p)$*. The sign of p ('+' if p is a start-point, '−' otherwise) is denoted by* $\mathrm{sign}(p)$*.*

Employing the idea of event-point sequences for events, we introduce the concept of *temporal arrangements* for event templates, which will be used in our pattern definition later on.

Definition 4.8 *(**Temporal Arrangement**) Let F be a tuple of n event templates, and Ω be a set of ET-points, defined as $\Omega = \{f^+ \mid f \in F\} \cup \{f^- \mid f \in F\}$.*

*A **temporal arrangement (TA)** of the tuple F is a sequence $S = p_1 \rhd p_2 \rhd .. \rhd p_{2n}$, where $p_i \in \Omega$, $\forall i \in \{1, .., 2n\}$.*

*A temporal arrangement S of a tuple of event templates F is called a **valid temporal arrangement** iff for each event template $f \in F$, f^+ occurs before f^- in the sequence S.*

As an example, the sequence $f_2^+ \rhd f_1^+ \rhd f_1^- \rhd f_2^-$ is a valid temporal arrangement for ETs f_1 and f_2, whereas the sequence $f_2^+ \rhd f_1^- \rhd f_1^+ \rhd f_2^-$ is not (because f_1^- occurs before f_1^+).

Different from event-point sequences, there are many possible TAs that are valid with respect to a set of event templates. For example, for a set of two ETs f_1 and f_2, there are six valid TAs corresponding to six relations BEFORE, AFTER, DURING, CONTAINS, OVERLAP, OVERLAPPED_BY, e.g., $f_1^+ \rhd f_1^- \rhd f_2^+ \rhd f_2^-$ represents [f_1 BEFORE f_2], or $f_1^+ \rhd f_2^+ \rhd f_2^- \rhd f_1^-$ represents [f_1 CONTAINS f_2].

From a syntactic point of view, Definition 4.8 allows one to define a valid temporal arrangement describing (temporal) relationships among event templates. However, such a temporal arrangement might not be (statistically) *significant*. For example, a TA describing [*workshop-sessions AFTER a-data-mining-conference*] might be less significant than a TA describing [*workshop-sessions BEFORE a-data-mining-conference*]. To determine how significant a temporal arrangement is, we introduce the concept of *instances* for temporal arrangements.

Definition 4.9 *(**Temporal Arrangement Support**) Let $F = \langle f_1, f_2, ..., f_n \rangle$ be a tuple of n ETs, and $S = p_1 \rhd p_2 \rhd \ ... \rhd p_{2n}$ be a temporal arrangement of F. Assume $f_i \prec f_j$, $i < j (\forall i, j \in \{1, 2, ..., n\})$.*

*A tuple of n events $E = \langle e_1, e_2, ..., e_n \rangle$, with the event-point sequence $T = q_1 \rhd q_2 \rhd ... \rhd q_{2n}$, **supports** S, or E is an **instance** of S, iff the following conditions hold*

1. *the event e_i is an instance of the ET f_i, $\forall i \in \{1, 2, ..., n\}$,*
2. *event(p_i) is an instance of ET(q_i), $\forall i \in \{1, 2, ..., 2n\}$, and*
3. *sign(p_i)= sign(q_i), $\forall i \in \{1, 2, ..., 2n\}$.*

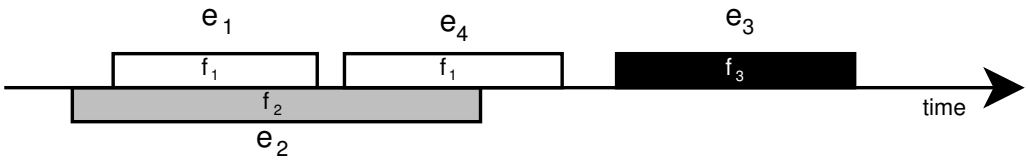


Figure 4.5: Example of a temporal arrangement of four events.

Generally speaking, the first condition states that each event in E is an instance of an ET in F, and the last two require that the sequences S and T describe the same temporal relationships.

To get a better understanding of the above concepts, consider Figure 4.5, which illustrates the relationships of time intervals for four events e_1, e_2, e_3, and e_4. Assume the events e_1, e_4 are instances of an ET f_1, the event e_2 is an instance of an ET f_2, and the event e_3 is an instance of an ET f_3. We then have:

- $\langle e_1, e_2 \rangle$ supports $S_1 = f_2^+ \rhd f_1^+ \rhd f_1^- \rhd f_2^-$ (f_1 DURING f_2), but $\langle e_4, e_2 \rangle$ does not,
- $\langle e_1, e_3 \rangle$ and $\langle e_4, e_3 \rangle$ both support $S_2 = f_1^+ \rhd f_1^- \rhd f_3^+ \rhd f_3^-$ (f_1 BEFORE f_3),
- $\langle e_1, e_2, e_3 \rangle$ supports $S_2 = f_2^+ \rhd f_1^+ \rhd f_1^- \rhd f_2^- \rhd f_3^+ \rhd f_3^-$ ((f_1 DURING f_2)BEFORE f_3).

4.5 Pattern Definition

Employing the concepts and definitions introduced in the previous sections, we now define so-called *Interval-based Event Sequence Patterns* (IESPs). Such patterns are to be discovered from a given dataset of events. We also provide a measure to determine the interestingness of patterns.

4.5.1 Interval-based Event Sequence Patterns

Interval-based Event Sequence Patterns, or IESPs for short, are patterns representing spatio-temporal proximity and conceptual relatedness, as well as temporal arrangements of events. Instances of a pattern are tuples of events supporting the pattern. We give formal definitions of patterns and instances as follows.

Definition 4.10 (IESP) *Given a tuple of event templates F ($|F| \geq 2$) and a temporal arrangement S of the set F, a pair $\langle F, S\rangle$ is called an **interval-based event sequence pattern**, or **IESP**, for short.*

Given an IESP $P = \langle F, S\rangle$, the size of the pattern P is the number of event templates in F.

For example, a pattern [*(f_1 overlap f_2) before f_3*], where f_1, f_2, and f_3 are some ETs, is represented as a size-3 pattern $\langle\langle f_1, f_2, f_3\rangle, f_1^+ \rhd f_2^+ \rhd f_1^- \rhd f_2^- \rhd f_3^+ \rhd f_3^-\rangle$.

Definition 4.11 ***(IESP Instance)*** *Given an IESP $P = \langle F, S\rangle$, with $F = \langle f_1, f_2, ..., f_n\rangle$ ($f_i \prec f_j$, $\forall i < j$, $i, j \in \{1, 2, ..., n\}$) and a temporal arrangement $S = p_1 \rhd p_2 \rhd ... \rhd p_{2n}$. A tuple of n events $E = \langle e_1, e_2, ..., e_n\rangle$ is a **row-instance** of P iff the following conditions hold*

1. *the events of E forms an event clique,*
2. *e_i is an instance of $f_i, \forall i \in \{1, 2, ..., n\}$, and*
3. *E supports S.*

This definition provides us with the basis for introducing an interestingness measure for an IESP, or, in other words, to determine which patterns are significant.

4.5.2 Interestingness Measure

A meaningful IESP is a pattern whose number of instances is statistically significant. Huang et al. [51] propose the *participation ratio* and *participation index* measures to determine whether or not a pattern (in their case co-location pattern) is statistically significant. These measures are computed from the number of events of a given event-type participating in the pattern and the total number of events having this type in the dataset. These measures work well if event-types have significant numbers of instances but they fail if some event-types are rare, as discussed in [52]. Furthermore, these measures do not take concept hierarchies of entities (objects forming co-location instances) into account. Since in most knowledge base datasets, concept hierarchies play an important role, we cannot directly apply the above measures for our approach and, therefore, have to do some adjustments.

Before giving formulas, we introduce the notion of *a table-instance* of an IESP $P = \langle F, S\rangle$, denoted $\mathcal{T}(P)$. Such a table consists of all row-instances of the pattern P, and each column corresponds to an ET $f \in F$. We use a notation $\mathcal{T}(P)[f]$ to

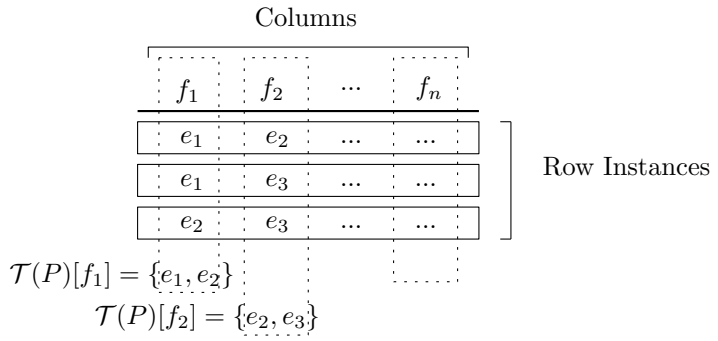


Figure 4.6: Example of a table-instance of a pattern $P = \langle\{f_1, f_2, ..., f_n\}, S\rangle$. Here, e_1, e_2, e_3 are events. The sets $\mathcal{T}(P)[f_1]$ and $\mathcal{T}(P)[f_2]$ consist of events participating in columns corresponding to f_1 and f_2, respectively.

denote the set of instances participating in $\mathcal{T}(P)$ at the column corresponding to the ET f. Figure 4.6 illustrates the concepts of table-instances, row-instances, and columns. One can see that an event can participate more than once in different row-instances, since an event may be a member of more than one event clique. A *participation ratio* is now defined as follows.

Definition 4.12 *(**Participation Ratio**) Given an IESP P, the **participation ratio** of an event template $f \in P$ is determined as*

$$pr(P, f) := \frac{|\mathcal{T}(P)[f]|}{|\{e : e \Vdash g,\ \forall g \in f^{\Uparrow}\}|} \qquad \in [0, 1]. \tag{4.1}$$

Generally speaking, the value of $pr(P, f)$ is the ratio of the number of events participating in a row-instance of the pattern P (in the column corresponding to the ET f) to the total number of events related to the ET f in the input dataset. For this, we enumerate all events that are instances of any generalization of the ET f. For example, if f is an ET with a context related to *'heavy metal'*, then all events related to *'music'* are enumerated, assume that the concept *'music'* is at the top level of the underlying hierarchy, and we do not care about the time and location components.

With the proposed formula, the more related events to an ET f participate in the table instance of P, the larger $pr(P, f)$ is. Based on participation ratios, the *prevalence* of a pattern is then the minimum value of all participation ratios.

Definition 4.13 (Prevalence) *The **prevalence** of an IESP P is determined as*

$$prev(P) := \min_{\forall f \in P}\{pr(P, f)\}. \tag{4.2}$$

*Given a threshold $\delta \in [0, 1]$, P is called **prevalent** if $prev(P) \geq \delta$.*

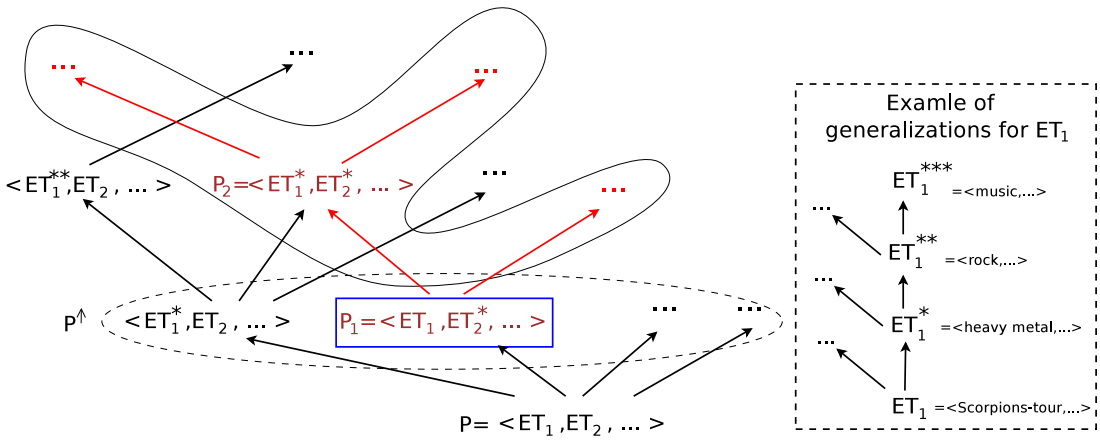


Figure 4.7: Generalizations of a pattern P= $\langle\{ET_1, ET_2, ...\}, S\rangle$. The set $P^{\uparrow}$ is marked by the dashed line. Assume that the pattern P_1 is the first prevalent pattern generated from P, marked by the box. Patterns derived from the pattern P_1 are marked within the solid line.

4.5.3 Most Specialized Prevalent Patterns

The prevalence measure is obviously useful to prune infrequent patterns during candidate generation. However, if a pattern is prevalent with respect to this measure and a given threshold, then a pattern derived from it by generalizing some components (ETs) is also prevalent with respect to the same threshold. Thus, the latter pattern is included in the resulting set, and it is then considered redundant. To address this problem, we introduce the concept of *redundant patterns* by formulating the generalization operators ($\uparrow$ and $\Uparrow$) for patterns, as detailed below.

Given a pattern P, $P^{\uparrow}$ is the set of patterns obtained from P by generalizing only one ET in P, i.e., replacing ET by an element in $\text{ET}^{\uparrow}$. As an example, Figure 4.7 depicts a lattice of patterns generated from a pattern P, where the set $P^{\uparrow}$ is marked by the dashed line.

Analogously, the set $P^{\Uparrow}$ consists of all patterns obtained from a pattern P by generalizing any ETs of P, i.e., replacing any ET by an element in $\text{ET}^{\Uparrow}$. In the example in Figure 4.7, the set $P^{\Uparrow}$ consists of all the patterns (excluding P) in the lattice.

We now specify (hierarchical) relationships between two patterns to determine which one is redundant.

Given two patterns P and Q, the pattern P is said to be *more generalized than* Q (or Q is *more specialized than* P) if and only if $P \in Q^{\Uparrow}$. If a result set contains both P and Q, it has a *redundant pattern*. Since the generalized pattern P can be inferred from the specialized one Q, P is a *redundant pattern*.

Definition 4.14 *(**Redundant Pattern**) Given a set of patterns $\mathcal{SP}$ and a pattern $P \in \mathcal{SP}$, the pattern P is called **redundant** in $\mathcal{SP}$ iff there exists a pattern $Q \in \mathcal{SP}$ such that $P \in Q^{\Uparrow}$.*

The definition above allows one to remove redundancies and keep only the most specialized ones in a pattern set. We use the term *most specialized prevalent patterns* or *MSPPs*, for short, to refer to prevalent patterns that do not have any prevalent specializations.

Definition 4.15 *(**Most Specialized Prevalent Patterns**) A pattern P is a **most specialized prevalent pattern (MSPP)** iff the following conditions hold*

1. *P is prevalent, and*
2. *there exists no prevalent pattern Q such that $P \in Q^{\Uparrow}$.*

To avoid the computation of redundant patterns, we eventually include only MSPPs in result pattern sets.

In Figure 4.7, we assume the pattern P_1 marked by the box is the first prevalent pattern generalized from the pattern P, i.e., P_1 is a MSPP. Let P_2 be a pattern derived from the pattern P_1, i.e., P_2 is one of the patterns in the set marked by the solid line. Obviously, every event tuple supporting P_1 also supports P_2, and since P_1 is prevalent, P_2 is also prevalent. However, P_2 is not a MSPP because P_1 is a prevalent pattern.

Exploiting the idea from the above example, if hierarchies for concepts, time, and locations are considered from the lowest levels to the highest ones, one can efficiently eliminate branches of a pattern lattice that cannot produce MSPPs. This will be described in detail in the next section.

4.6 Mining Most Specialized Prevalent Patterns

Like for mining spatio-temporal patterns from "traditional" datasets, mining such patterns from knowledge base datasets is computationally expensive. This is due to not only the large number of events in the datasets but also because of the oftentimes large concept hierarchies. Therefore, existing methods for generating candidate patterns from the set of all possible ETs (i.e., by incrementally replacing contexts, time, locations of events by their direct or indirect generalizations in respective hierarchies) are very inefficient or even infeasible.

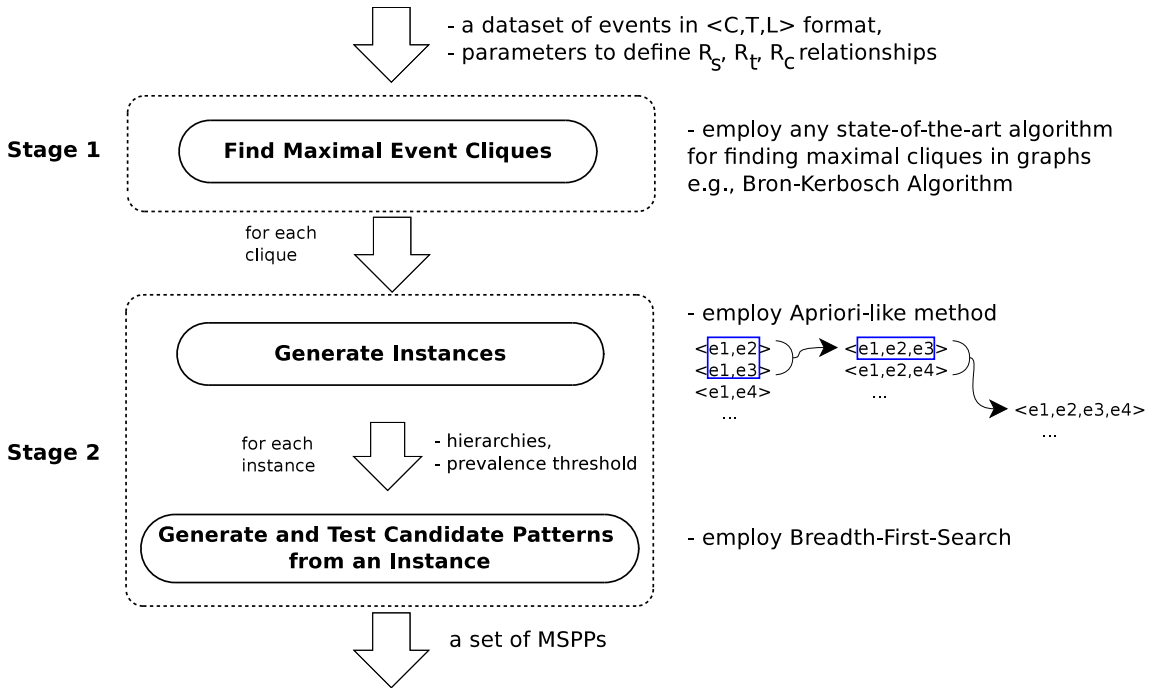


Figure 4.8: Overview of the proposed approach to mine MSPPs.

In our studies, we observed that the number of events in a dataset is much smaller than the number of ETs that can be generated. Thus, in our approach, we start with instances and then generate patterns from combinations of such instances. Since an instance is required to be an element of at least one event clique, enumerating instances can be done by finding maximal event cliques. Figure 4.8 gives an overview of our approach consisting of two stages to mine MSPPs. The first stage is to find maximal event cliques, and the second one is to generate and test candidate patterns. These stages are detailed as follows.

The first stage is to find *maximal event cliques*. For this, an undirected graph is built from the set of events where each event is a vertex and two events share an edge if they have spatial, temporal, and conceptual relationships R_s, R_t, and R_c, respectively. Any state-of-the-art algorithm for finding maximal cliques in graphs, such as the *Bron-Kerbosch* algorithm [14] or its variants (e.g., [108]), can be employed here. We omit respective details because this is not the focus of our work. For enumerating maximal event cliques, we simply assume a function called Compute_Maximal_Clique from an event dataset, as shown in Line 2 of Algorithm 4.1.

In the second stage, each maximal clique is examined to generate MSPPs in two steps: (1) generating instances from the current clique, and (2) generating patterns for each created instance, as shown in Figure 4.8 and also in Lines 3-20 of Algorithm 4.1.

Algorithm 4.1: MSPP-Miner

Input:
(a) D: an event dataset
(b) $\mathcal{H}$: a set of hierarchies (for context, time, location)
(c) R_s, R_t, R_c: spatial, temporal, and conceptual relationships
(d) δ: a prevalence threshold
Output: a set of all MSPPs

```
1  result = {};
2  CliqueSet=Compute_Maximal_Cliques(D, R_s, R_t, R_c);
3  foreach Clique C in CliqueSet do
4      L_1 = { ⟨e⟩ | ∀e ∈ C};// a set of size-1 instances;
5      k = 1;
6      while L_k ≠ {} do
7          L_{k+1} = {};
8          foreach Instance I_k ∈ L_k do
9              foreach Instance I'_k ∈ L_k do
10                 I_{k+1} = I_k ⋈ I'_k; // join-operator combining two size-k instances ;
11                 if I_{k+1} ≠ null then
12                     Patterns_from_I_{k+1}={};
13                     Compute I^↑_{k+1} from I_{k+1} and H;
14                     foreach Candidate Pattern P ∈ I^↑_{k+1} do
15                         L=Search_In_Lattice(P, δ, H);
16                         Patterns_from_I_{k+1} = Patterns_from_I_{k+1} ∪ L;
17                     Remove redundancies in Patterns_from_I_{k+1};
18                     if Patterns_from_I_{k+1} ≠ {} then
19                         L_{k+1} = L_{k+1} ∪ {I_{k+1}};
20                         result = result ∪ Patterns_from_I_{k+1};
21         k = k + 1;
22 return result;
```

Algorithm 4.2: Search_In_Lattice(P, δ, $\mathcal{H}$)

```
   /* return a set of all MSPPs having paths on the lattice from P     */
1  result={}; Queue=[P];
2  while Queue is not empty do
3      P'=Queue.dequeue();
4      Compute the set P'^↑ from P' and H;
5      foreach Pattern Q ∈ P'^↑ do
6          if Q is prevalent with respect to δ then
7              result = result ∪ {Q};
8          else
9              Queue.enqueue(Q);
10 return result;
```

For the step of generating instances, instead of inefficiently generating all subsets of each clique, we employ an Apriori-like method that works as follows. We start generating a set of size-1 instances L_1 from the current clique C, that is, each event in C becomes an instance in L_1 (Line 4). For $k \geq 1$, size-(k+1) instances are computed by joining two size-k instances in L_k (Line 10). Similar to the join-step in any Apriori-like algorithm [43], joining an instance $I_k = \langle e_1, e_2, ..., e_k \rangle$ ($e_i \prec_e e_j,\ 1 \leq i < j \leq k$) with an instance $I'_k = \langle e'_1, e'_2, ..., e'_k \rangle$ ($e'_i \prec_e e'_j,\ 1 \leq i < j \leq k$), denoted $I_k \bowtie I'_k$, works as follows: if ($e_i = e'_i,\ 1 \leq i \leq k-1$) and ($e_k \prec_e e'_k$) then $I_k \bowtie I'_k = \langle e_1, e_2, ..., e_k, e'_k \rangle$, otherwise $I_k \bowtie I'_k = null$. Here we use *null* to indicate no size-($k+1$) instance is generated from I_k and I'_k, or the instances I_k and I'_k cannot be joined.

For each new instance I_{k+1}, the set of MSPPs that can be derived from I_{k+1} is computed by using Breadth-First-Search on the lattice of patterns (Line 13-16). Such a lattice is generated using the hierarchies for the components involved in the instance as shown in Algorithm 4.2. One can see that when in the lattice a node with respect to a pattern Q is visited, the prevalence of Q is computed (using Equation 4.2). If Q is prevalent then Q is a MSPP and, therefore, all generalizations of Q can be ignored. Otherwise, the routine continues searching in $Q^{\uparrow}$.

Results of Algorithm 4.2 are used in Lines 15 and 16 of Algorithm 4.1 to compute a set of all MSPPs from a size-(k+1) instance. All redundant patterns (Definition 4.14) in this set are removed in Line 17. Next, all size-($k+1$) instances that can produce MSPPs are included in L_{k+1} for the next iteration (Line 19). Algorithm 4.1 stops if no more new instances are created, i.e., $L_{k+1} = \{\}$.

The runtime of our approach heavily depends on the size and the characteristics of the input dataset. In particular, the more candidate patterns are generated (or the more nodes in the pattern lattice are visited), the more CPU time is consumed. In the worst-case, all nodes of the pattern lattice are visited, and these can be many. However, the number of candidate patterns generated in practice is often much smaller than the size of the pattern lattice. Therefore, similar to other pattern mining approaches that employ the Apriori method, we only analyze the experimental runtime of Algorithms 4.1 and 4.2 for real datasets, as shown in the next section.

4.7 Experimental Evaluation

We demonstrate the utility of our approach using two datasets. The first dataset is extracted from YAGO2, and the second one is crawled from the Website *eventful.com*. Our objective is to show how to apply the framework to mine interesting patterns

from different datasets. In the following, we describe the setup of our experiments, and we present some interesting patterns extracted from each dataset. Our framework was implemented in Java and run with 24GB heap size. All experiments were run on an Intel Xeon 2.27GHz with 48GB RAM, running Ubuntu 64bit. We describe in order the experiments for each dataset in the following sections.

4.7.1 YAGO2 Dataset

Dataset Description

YAGO2 [48] is known as a large knowledge base extracted from Wikipedia in which each fact is described as a subject-predicate-object triple. To employ our event model, such a triple is considered an event context.

In YAGO2, some facts are also anchored in space and time. Since we are interested in not only the context, but also in time and location components of events in finding patterns, we consider only facts for which both spatial and temporal components are specified. For example, in YAGO2, facts describing historical events use the *'happenedOnDate'* predicate. We selected such facts to demonstrate our framework. We also extracted a concept hierarchy from the facts that use the *'type'* and *'subClassOf'* predicates. To prevent meaningless patterns, and also to reduce the size of the concept hierarchy we excluded some categories that are too general, e.g., *wordnet_object_10002684* can be a physical object, a property, or a work of art.

Through that procedure, we finally obtained a dataset, called HOD-YG (*happenOnDate*-facts from YAGO2), consisting of 1,869 facts (events) and a hierarchy of 27,462 concepts used for subjects in the events. Table 4.1 shows the number of concepts at each level, where the level of a concept is specified by Definition 3.4. One can see that Level 0 of the hierarchy contains entities corresponding to the subjects of the facts. Levels 1 and 2 consist of categories originating from either Wikipedia or WordNet. Most of concepts are at Level 1, and they are typically Wikipedia categories.

Since the events have the same *'happenedOnDate'* predicate and objects in the events are dates (the same values as time components), we omit predicates and objects in representing events and event templates. That is, the context of an event contains only the subject of the corresponding fact.

Experimental Setup

The HOD-YG dataset consists of historical events such as wars and battles, where the temporal granularity is typically 'Year'. Thus, we consider two levels of abstraction

Table 4.1: Distribution of concepts at different levels in the concept hierarchy for the dataset HOD-YG.

Level	Number of Concepts
0	1,869
1	25,472
2	121

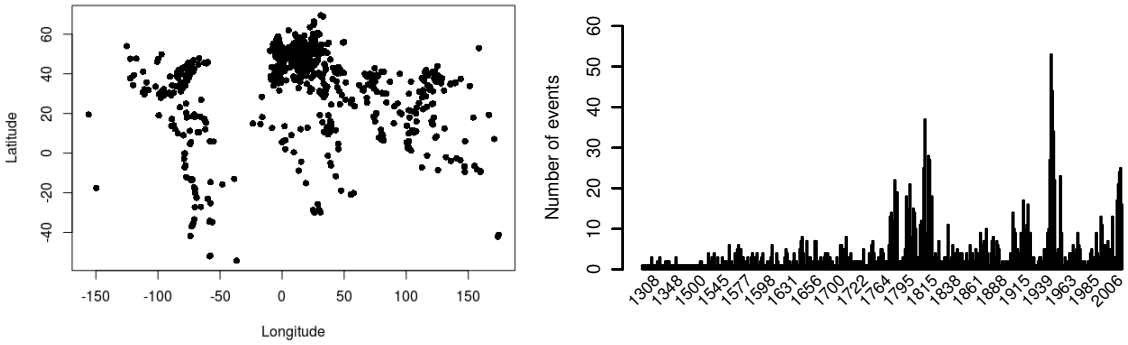


Figure 4.9: Spatial and temporal distributions of the HOD-YG dataset.

for generalizing time, i.e., $Year$ and All_{time}. Analogously, since most of the locations refer to countries, we consider $Country$ and All_{loc} for generalizing locations.

Since the selected facts from YAGO2 are historical events like wars, battles, conflicts, etc., they are considered conceptually related. We consider only spatial and temporal proximity (i.e., R_s, R_t relationships) for events to compute maximal event cliques. Figure 4.9 illustrates the spatial and temporal distributions of the HOD-YG dataset. In our experiments, the spatial relationship R_s is defined using the Euclidean distance and a distance threshold d. For the temporal relationship R_t, we set the value for the time window size t_w. To choose suitable values for d and t_w, we tried different settings, ranging from 1 to 10 km, for the distance threshold, and from 1 to 5 years, for the time window size. Basically, the smaller the thresholds, the more interesting the patterns should be. However, no pattern or only size-2 patterns will be found when low threshold values are used. Since changing values for these thresholds affects the result of finding maximal event cliques, one can select suitable values for d and t_w by sampling maximal event cliques from the result. For the HOD-YG dataset, as shown in experimental results below, when d is set to 1 km, and t_w is set to 2 years, we obtain some interesting patterns.

For the prevalence threshold δ, we run the algorithm with different settings of δ ranging from 0 to 1. Obviously, the smaller the threshold, the more patterns are found. When $\delta > 0.5$, no pattern is found and when $\delta \leq 0.01$, some patterns with

Table 4.2: The number of patterns and the runtime for the dataset HOD-YG.

Dataset	δ	Candidates	Patterns	Pattern Sizes	Runtime
	0.02	15,405	641	2-8	82 seconds
HOD-YG	0.05	102,465	174	2-6	557 seconds
(1,869 events,	0.1	143,228	54	2-6	792 seconds
231 max.cliques)	0.2	19,207	12	2-3	14 seconds
	0.3	6,347	5	2	11 seconds

Table 4.3: Sample patterns extracted from the HOD-YG dataset. All_{time} and All_{loc} are denoted by * . The values in [ ] represent participation ratios of ETs in the patterns.

Some patterns extracted from HOD-YG	Prev.
Y1. {wordnet_conflict.*.*(0), wordnet_battle.*.*(1)} $0^+ \rhd 1^+ \rhd 0^- \rhd 1^-$ [pr(0):515/1818, pr(1):467/1363]	0.28
Y2. {wordnet_conflict.*.*(0), wordnet_battle.*.*(1)} $1^+ \rhd 0^+ \rhd 1^- \rhd 0^-$ [pr(0):232/1818, pr(1):227/1363]	0.13
Y3. {wordnet_conflict.*.*(0), wordnet_battle.*.*(1)} $0^+ \rhd 0^- \rhd 1^+ \rhd 1^-$ [pr(0):171/1818, pr(1):129/1363]	0.09
Y4. {wordnet_conflict.*.*(0), wordnet_battle.*.*(1)} $1^+ \rhd 1^- \rhd 0^+ \rhd 0^-$ [pr(0):144/1818, pr(1):153/1363]	0.08
Y5. {wikicategory_NATO_operations.*.Afghanistan(0), wordnet_operation.*.Afghanistan(1) $1^+ \rhd 1^- \rhd 0^+ \rhd 0^-$ [pr(0):7/96, pr(1):8/96]	0.07
Y6. {wikicategory_Naval_battles_involving_pirates.*.Somalia(0), wordnet_incident.*.Somalia(1)} $0^+ \rhd 1^+ \rhd 0^- \rhd 1^-$ [pr(0):8/225, pr(1):6/38]	0.04

only few instances are discovered. For example, the number of patterns and runtime for different δ are shown in Table 4.2.

This table also illustrates the relationships between the runtime and the number of generated candidates. One can see that the number of generated candidates heavily depends on a particular value of δ. For example, the number of candidates when $\delta = 0.1$ is about nine (eight) times larger than when $\delta = 0.02$ ($\delta = 0.2$). Recall that all direct generalizations of a candidate P will be new candidates for the next iteration if the candidate P does not satisfy the threshold δ. Thus, increasing the value of δ might not decrease the number of the total candidates generated. However, increasing the value of δ will clearly decrease the maximum size of candidate patterns. As shown in Table 4.2, although the number of candidates when $\delta = 0.2$ is larger than when $\delta = 0.02$, the runtime of the first case is about six times smaller than the runtime of the latter case. The reason is that in the first case, only size-2 and size-3 candidates are generated, whereas candidates in the latter case are generated up to size-8.

Experimental Results

We obtained from the HOD-YG dataset about 380 patterns, where the prevalence of each pattern is no less than $\delta = 0.02$. Most of these patterns relate to battles,

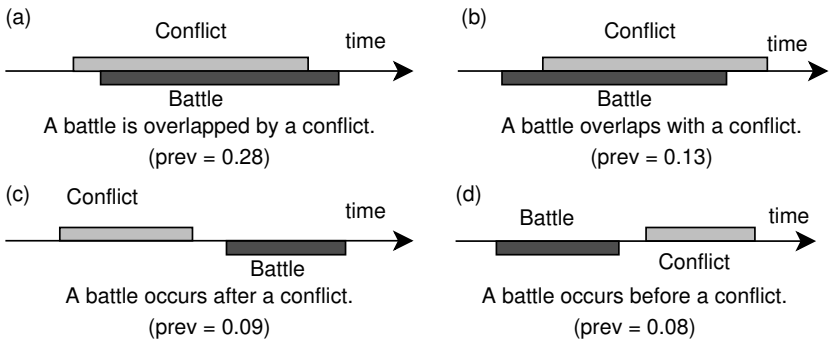


Figure 4.10: Illustrations of patterns Y1(a), Y2(b), Y3(c) and Y4(d) in Table 4.3.

wars, or conflicts like the patterns shown in Table 4.3. Since ETs are obtained by generalizing subjects, time, and locations, we represent ETs of patterns in the form of *subject.time.location*. The pattern Y1, for example, can be interpreted as: *"28% (515/1818) of conflicts are nearby to and overlaps a battle"* and *"34% (467/1363) of battles are nearby to and overlapped by a conflict"*. The patterns Y2, Y3, Y4 are similar to the pattern Y1 in terms of the topics *'conflict'* and *'battle'*, but they describe different temporal relationships, as shown in Figure 4.10. Among these patterns, the first two (Y1 and Y2) are more prevalent than the other ones (Y3 and Y4). Thus, one can infer that the two concepts *'conflict'* and *'battle'* are closely related.

Besides, we also found other interesting patterns such as patterns that describe relationships between pirates and incidents in Somalia, or between NATO operations and other military actions in Afghanistan, as shown in Table 4.3.

We also found some patterns consisting of more than three event templates. However, in such patterns, all temporal relationships are of type OVERLAPS and the elements are the same, for example, *"a battle overlaps two other battles"* or *"a conflict overlaps two other conflicts"*.

4.7.2 Eventful Dataset

Dataset Description

We use another dataset, called EVF, crawled from the Website *eventful.com* to demonstrate the feasibility and utility of our framework. This Website provides services for users to find, track, and share information about events. Each event consists of an event identifier, title, time, location, and a list of tags. Tags are keywords attached to events to categorize events. For our experiments, the EVF dataset contains about 7 million events from 2009 to 2013 for many different topics such as music, sports,

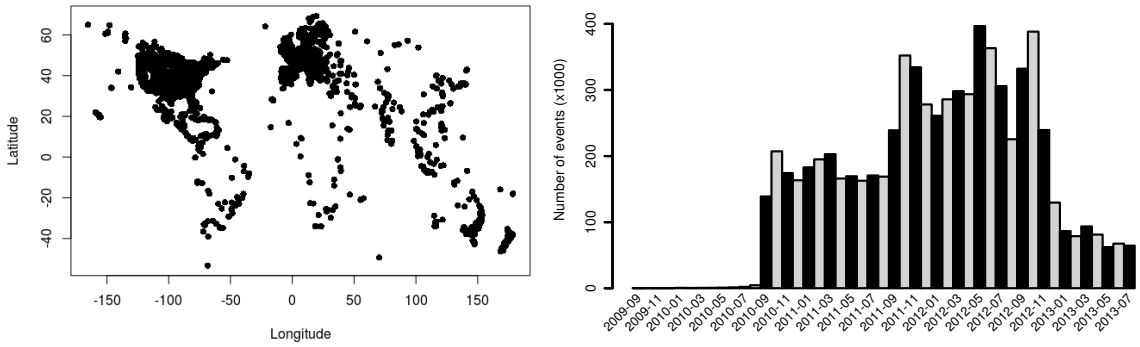


Figure 4.11: Spatial and temporal distributions of the EVF dataset.

and politics. Figure 4.11 shows the spatial and temporal distributions of this dataset where most of the events are located in the US and Europe in three years, 2010, 2011, and 2012.

Experimental Setup

For efficiently computing maximal event cliques, we considered conceptual constraints (R_c) in a data preprocessing step. That is, we selected several topics from the EVF dataset and for each topic, we created a sub-dataset. For example, to mine patterns related to the topic *'election'*, a sub-dataset is created by selecting all events whose tag list contains *'election'*. Such a dataset is an input for the mining algorithm.

Similar to the dataset YAGO2, we considered each event as a spatio-temporal fact, where the predicate is *'happenedOnDate'*, the subject is the event identifier and the object is the start-time. Each tag is also considered a category of events.

In a data pre-processing step, tags were stemmed and all stop words were removed. We computed the *sub-class-of relationships* between two tags by using the following assumption: a tag T_1 is a subclass of a tag T_2 if the set of events having tag T_1 is a subset of the set of events having tag T_2. For example, from the same event dataset, the set of events related to *'jazz'* is obviously a subset of the set of events related to *'music'*. From the directed graph where nodes are tags and edges represent *sub-class-of relationships*, we eliminated transitive links. For example, *'heavy metal'*→*'music'* is removed since we have two links: *'heavy metal'*→*'rock'* and *'rock'*→*'music'*.

Table 4.4 shows the number of concepts at each level for three hierarchies used for datasets EVF-EL, EVF-GM, and EVF-CP that are extracted for three topics *'election'*, *'music in Germany'*, and *'charity & politics'*, respectively.

Table 4.4: Distributions of concepts at different levels in the concept hierarchy for the datasets EVF-GM (music in Germany), EVF-CP (charity & politics), and EVF-EL (election).

Level	**Number of Concepts**		
	EVF-GM	EVF-CP	EVF-EL
0	2,583	4,228	7,977
1	5,109	3,903	4,788
2	1,007	1,161	1,093
3	80	85	69
4	14	14	12

Table 4.5: Statistics of experiment results for the datasets extracted from the EVF dataset. The prevalence threshold δ is set to 0.1 for each dataset.

Dataset	**Events**	**Max.Cliques**	**Candidates**	**Patterns**	**Pattern Sizes**	**Runtime**
EVF-GM	2,583	718	276,600	227	2-5	3 mins
EVF-CP	4,228	819	343,883	207	2-6	8 mins
EVF-EL	7,977	2,615	1,381,409	268	2-4	261 mins

Based on the spatial and temporal granularities for time and locations of events in the EVF dataset, we specify the time and location hierarchies as $(Day \rightarrow Month \rightarrow Year \rightarrow All_{time})$ and $(City \rightarrow State \rightarrow Country \rightarrow All_{loc})$, respectively.

Experimental Results

To demonstrate that not only the number of events but also the number of maximal cliques affects the runtime of the mining process, we present results from the three datasets EVF-EL, EVF-GM, and EVF-CP as described above. The number of events and the number of maximal cliques for each dataset is presented in Table 4.5. One can see that the dataset EVF-EL is the largest in terms of the number of events and the number of maximal cliques.

Similar to the method described for the experiments using the YAGO2 data, we tried different settings for the parameters. Based on the analysis of the maximal event cliques, we finally selected 1 km for the distance threshold and 3 days for the time window t_w. Using these settings, we obtained interesting patterns, as presented in the following.

Table 4.5 summarizes the experimental results. One can see that the runtime closely related to the number of generated candidates. Obviously, the number of generated candidates depends on characteristics of the input dataset, including the number of maximal cliques, and the number of events and event templates in each maximal clique. Accordingly, among the three datasets, the runtime of the dataset EVF-EL is largest, as shown in Table 4.5.

Table 4.6: Selected patterns extracted from the EVF dataset. All_{time} and All_{loc} are denoted by *. The values in [ ] represent participation ratios of ETs in the patterns.

Pattern	**Prev.**
Some patterns extracted from EVF-EL	
E1. {barackobama.2010.US(0), politics.2010.US(1)} $0^+ \rhd 0^- \rhd 1^+ \rhd 1^-$ [pr(0):2692/7546, pr(1):2693/7878]	0.34
E2. {barackobama.2010-10.US(0), dnc.2010-10.US(1)} $0^+ \rhd 0^- \rhd 1^+ \rhd 1^-$ [pr(0):2630/7546, pr(1):2656/7546]	0.35
E3. {campaign.2010.US(0), politics.2010.US(1), barackobama.2010.US(2)} $0^+ \rhd 0^- \rhd 1^+ \rhd 2^+ \rhd 1^- \rhd 2^-$ [pr(0):987/7878, pr(1):980/7878, pr(2):980/7546]	0.12
Some patterns extracted from EVF-GM	
G1. {music.*.Berlin-DE(0), berlin.*.Berlin-DE(1)} $1^+ \rhd 1^- \rhd 0^+ \rhd 0^-$ [pr(0):434/2580, pr(1):430/1256]	0.17
G2. {show.2010-12.Neukoelln-Brandenburg-DE(0), christmas.2010-12.Neukoelln-Brandenburg-DE(1)} $1^+ \rhd 1^- \rhd 0^+ \rhd 0^-$ [pr(0):19/155, pr(1):19/58]	0.12
G3. {music.*.DE(0), rock.*.DE(1), jazz.*.DE(2)} $0^+ \rhd 0^- \rhd 1^+ \rhd 1^- \rhd 2^+ \rhd 2^-$ [pr(0):454/2580, pr(1):258/2580, pr(2):262/2580]	0.10
Some patterns extracted from EVF-CP	
C1. {volunteer.2011.SDiego-Cali-USA(0), community services.2011.SDiego-Cali-USA(1)} $0^+ \rhd 1^+ \rhd 0^- \rhd 1^-$ [pr(0):313/1187, pr(1):301/1182]	0.25
C2. {politics.*.US(0), community.*.US(1)} $1^+ \rhd 1^- \rhd 0^+ \rhd 0^-$ [pr(0):605/4226, pr(1):589/3631]	0.14
C3. {nonprofit.2011.Cali-US(0), volunteer.2011.Cali-US(1)} $1^+ \rhd 0^+ \rhd 0^- \rhd 1^-$ [pr(0):419/4102, pr(1):279/1187]	0.10

From the Table 4.5, one can see that the sizes of discovered patterns are from 2 to 6. We present only some typical ones for each dataset in Table 4.6.

In EVF-EL dataset, since most of events relate to the president of the United States, Barack Obama, many patterns related to this president were found. One can see that patterns related to the president and *'politics'*, *'democratic'*, *'campaign'*, etc., have high prevalence values.

In the EVF-GM dataset, we found some interesting patterns related to Germany and music. For example, we interpret pattern G1 in Table 4.6 as: *"34% (430/1256) of the events in Berlin are nearby to and followed by a musical event"* and *"17% (434/2580) musical events in Berlin are nearby to and follow another event"*. Pattern G3 is an example of a size-3 pattern, representing a series of musical events like common for music festivals.

In the third dataset EVF-CP, patterns related to *'volunteer'*, *'nonprofit'*, etc. were found. Pattern C1, can be interpreted as follows: *"In 2011, 26% (313/1187) of the events related to volunteer in San Diego are nearby to and overlap with another event related to community services"* and *"In 2011, 25% (301/1182) of the events related to community services in San Diego are nearby to and overlapped by another event related to volunteer"*. Note that in most of the patterns related to *'volunteer'*, time and

locations are generalized to *'2011'* and *'San Diego-California-US'* because all *'volunteer'* events originally come from the Website *http://www.volunteersandiego.org*.

4.8 Discussion

Linked Open Data sources and respective knowledge bases will become a valuable source for mining spatial and spatio-temporal patterns from non-traditional data. In this chapter, we presented an approach to discover such patterns from facts describing events that consist of context, time, and location components. In particular, we showed how hierarchies associated with respective components of events can be exploited to derived interesting spatio-temporal patterns at different levels of granularity and abstraction. We demonstrated the feasibility and utility of our framework using real datasets.

Similar to other approaches to the discovery of frequent patterns, the pattern explosion problem also arises in our experiments, that is, the mining result often consists of massive amounts of prevalent patterns. Selecting the top most 'interesting' patterns from that result is not trivial since the interestingness of a pattern is clearly subjective. One obvious direction of future work is to investigate methods that summarize or compress the set of discovered patterns, that is, to find as small as possible subsets of (prevalent) patterns that can fully describe the characteristics of the original dataset.

Chapter 5

Mining Periodic Event Patterns

Periodic behaviors commonly exist in real-life event data, such as daily human activities, weekly TV programs, monthly meetings, seasonal sales, or annual festivals. The discovery of patterns exhibiting periodicities from such data is important for several applications, for instance, forecasting future events, predicting user behavior from social media data for service suggestion and product promotion, or identifying anomalies in data for fraud prevention.

Existing approaches to the discovery of periodic patterns mostly focus on traditional data such as time series, genome sequence data, or trajectories of moving objects. However, with the rapid growth of social media channels, numerous data sources managing information about events are now available. The presence of spatial, temporal, and conceptual relationships between events described in such data sources gives rise to new approaches for discovering interesting periodic patterns.

In this chapter, we present an approach aiming at the discovery of interesting periodic patterns in the presence of conceptual, temporal, and spatial hierarchies. Similar to Chapter 4, we utilize the event model introduced in Chapter 3 to formulate events and event topics in considering hierarchies for time, locations, and concepts associated with event descriptions. Periodic event patterns are then recurring patterns of event topics that co-occur and have temporal regularities. We demonstrate the feasibility and utility of our framework using real event datasets extracted as RDF facts from the Website eventful.com.

Some initial ideas and results presented in the following sections appeared in a paper by Le and Gertz [66]. In this chapter, we extend these ideas by providing: (1) lemmas and proofs for the method, (2) more elaborate explanations for the notations and definitions, (3) more comprehensive experiments on larger event datasets, and (4) discussions in more detail about the runtime and experimental results.

5.1 Introduction

Sequential pattern mining and periodic pattern mining are two problems aiming at the discovery of insights or knowledge from time-related data [43, 118]. While Chapter 4 focuses on the problem of sequential pattern mining, this chapter addresses the problem of the discovery periodic patterns from event data, where the user is interested in events co-occurring in some periodicity. Such patterns are useful in several applications, such as event prediction, service suggestion and product promotion for the user, or anomaly detection. For example, if a user searches for information about a music concert, and we know the fact that such a concert usually co-occurs with other events in some periodicity, e.g., the last Saturday every month, then these events might be good suggestions for that user at some suitable time.

Although many approaches have been proposed for the discovery of periodic patterns from time-related data, e.g., [45, 75, 85], existing approaches mostly deal with traditional sequence data, such as time series, symbolic sequences, or trajectories of moving objects. In these approaches, event relationships are simply specified on the basis of the order of events in a sequence. Different from these approaches, we aim at mining periodic patterns from event data sources, where relationships between events are not only a temporal order but also spatial and conceptual relationships. Beside traditional challenges (e.g., sparse, noisy data, or incomplete observations [73]), the presence of such relationships poses some new challenges. One of these challenges is that one needs to specify how events can be linked in a sequence by exploiting the temporal, spatial, and conceptual relationships between events for the purpose of detecting periodicities. Another challenge is that with each event component (the context, time, or location), ontologies are associated in which concepts are generalized and organized in hierarchies. Approaches to the discovery of periodic event patterns thus have to take such hierarchies into account to derive patterns at different levels of granularity and abstraction.

By employing the concepts and notations for events and event templates introduced in Chapter 3, we present a novel approach to the discovery of periodic event patterns from event data. That is, from a set of event instances, using conceptual, temporal, and spatial hierarchies, we want to determine (a) periodicities associated with event topics (specified by event templates) and (b) periodicities associated with co-occurring event topics. Such information can be used to make predictions about upcoming events, to detect outliers in a given dataset, or to explicitly add information about periodicity and co-occurrence to event descriptions.

In order to guide the search for interesting periodic patterns, we allow the user to specify constraints related to the conceptual, temporal, and geographic information associated with event descriptions. For example, from a large dataset, it might be trivial to derive that some music event is happing every Saturday (independent of the location), but if one is looking for patterns in music event (*'music'* being a generalization of music related events) that *periodically* occur in the same region or place, then such constraints can narrow down the search and allow for a more efficient discovery of patterns.

In summary, the contributions of this chapter are as follows:

- We model *periodic event patterns* where a pattern consists of event topics that co-occur and repeat over time. For this, we formulate periodicities for event topics by introducing the concepts of *time slots* and *segments*.
- We propose a suitable interestingness measure for the selection of significant periodic event patterns based on the concepts of *p-scores* and *event instance vectors* (EIV).
- We define constraints for events and event templates to find only patterns that satisfy conditions specified by the user.
- We propose an effective approach to the discovery of periodic event patterns.
- Finally, we demonstrate the feasibility and utility of our approach using a large-scale event dataset extracted as RDF facts from the Website eventful.com. We show that in fact interesting periodic patterns can be extracted from such event data, which then can be used, e.g., to enrich the existing RDF dataset by further facts.

In the following section, we discuss related work. In Section 5.3, we formulate event occurrences and constraints for events. In Section 5.4, we introduce a specification language for periodic patterns. We then describe our method to mine such patterns in Section 5.5. After presenting some experimental results in Section 5.6, we summarize this chapter in Section 5.7.

5.2 Related Work

Our work is closely related to approaches developed in (1) periodicity detection from symbolic sequences and time series data, (2) cyclic association rule mining from temporal databases, and (3) periodic pattern mining from spatio-temporal data.

In Section 2.2.3, we introduced existing approaches to automatically detect unknown periods for time series and symbolic sequences. Briefly, traditional approaches such as the Fast Fourier Transform or autocorrelation [114] can be utilized for sequences exhibiting full-cycle periodicities. Other approaches are proposed for specific cases, such as partial periodicity [42, 45, 75], asynchronous periodicity for noisy data [126], or periodicity for data with incomplete observation [73]. Although our focus is on (1) types of data different from those studied in the above approaches and (2) on periodic patterns representing co-occurrences of multiple event topics, we still can employ any of the above methods to detect periods for single events and then use this information to find periodic patterns.

The problem of mining cyclic association rules has been introduced by Özden et al. [85] where an association rule R is cyclic with respect to a positive integer p if the rule R holds every p time units. This approach aims at finding perfect periodicities and, therefore, it is impractical as patterns in real life are often imperfect [45]. Han et al. [42] consider partial periodic patterns where periodicities satisfying some confidence threshold are allowed. In general, all the above approaches are designed for time-related datasets, but not for datasets that also include spatial information or where explicit sequences of events do not exist.

In the context of moving object data, a trajectory of an object can be viewed as a sequence of geo-coordinates. Typically, a moving object rarely appears at exactly the same coordinate as in the previous cycle. Therefore, applying traditional approaches on such raw trajectories might not work. For example, although an individual uses the same walking route from his home to his office on a daily basis, he might start earlier than usual or stop somewhere, e.g., to buy a newspaper. Thus, to deal with trajectory data, existing approaches typically transform raw trajectory data into a less noisy form. Mamoulis et al. [77] propose an approach in which raw trajectories are transformed into sequences of regions, e.g., districts or mobile communication cells. Periodic patterns are then cyclic sequences of visited regions. A limitation of this approach is that it assumes a set of predefined regions to convert a trajectory to a symbolic sequence. Rather than assuming predefined regions, Li et al. [71] propose a method to detect important regions, called reference spots, using a kernel method, from which then periodic behaviors are determined.

Different from the approaches mentioned above, we focus on the discovery of periodic patterns from event data where explicit sequences of events do not exist. Furthermore, our approach is able to derive periodic patterns at different levels of

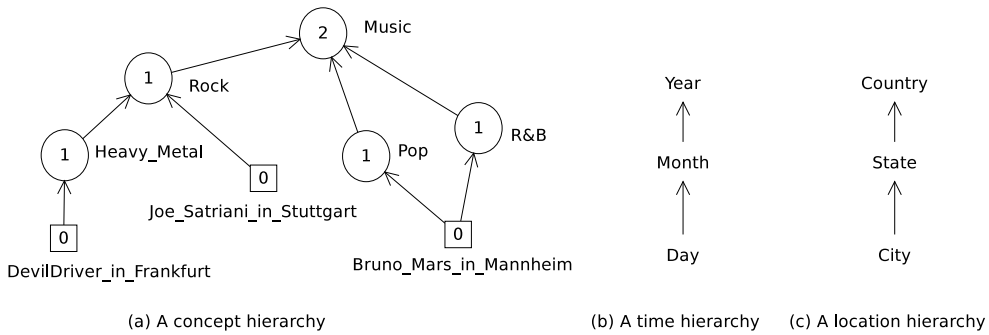


Figure 5.1: Example of hierarchies for concepts, time, and locations.

abstraction, using hierarchies associated with the description of event components, which are not considered by existing approaches.

Before presenting our approach, we introduce some basic concepts and notations in the following section.

5.3 Basic Concepts and Notations

To model periodic event patterns, we specialize the notations and definitions of events and event templates from the event model presented in Chapter 3. They are described in the following.

5.3.1 Event Occurrences and Constraints

Similar to formulating *interval-based event patterns* in the previous chapter, we employ the event model presented in Chapter 3 to define a pattern specification language for the discovery of *periodic event patterns*. While Chapter 4 focuses on interval-based relationships among events where event durations play an important role, this chapter primarily focuses on relationships among occurrences (not durations) of events. Thus, the time point framework described in Definition 3.17 (Section 3.3.2 - Page 41) is employed to represent event time.

By using our event model, an event is specified as a tuple $\langle e_{id}$, Context, Time, Location$\rangle$, where the first component is an event identifier, and the last three components can be generalized to higher levels of abstraction and granularity, based on underlying hierarchies, to obtain so-called *event templates (ETs)*.

As an example, the tuple "$\langle$#001, *DevilDriver_in_Frankfurt*, $\langle$*Day*,*'August 9, 2013'*$\rangle$, $\langle$*City*,*'Frankfurt'*$\rangle\rangle$" describes a music event with an identifier #001, on 9th of August 2013, in the city of Frankfurt. This event can be generalized to ETs, for instance,

$\langle$*Heavy_Metal*, $\langle$*Year*, *'2013'*$\rangle$, $\langle$*State*, *'Hesse'*$\rangle\rangle$, or
$\langle$*Music*, $\langle$*Month*, *'August, 2013'*$\rangle$, $\langle$*Country*, *'Germany'*$\rangle\rangle$,
by using the conceptual, temporal, and spatial hierarchies shown in Figure 5.1. Such event templates are key ingredients of patterns describing periodicities of event topics. The efficient discovery of these patterns is the focus of this chapter.

Typically, periodicities are determined from a sequence of events. However, such sequences do not explicitly exist in an event dataset as described above. Thus, for the purpose of mining periodic patterns from an event dataset, one needs to specify how events can be linked in a sequence. For this, we allow the user to specify temporal, spatial, and conceptual constraints on events, as introduced in Section 3.6.1 (Page 52). These constraints represent the requirement of the user for the discovery of periodic patterns. For example, in practice, users might be interested in patterns that exhibit periodicities in a specific time interval and/or in a specific geographic region and simply refer to a specific concept. Briefly, a sequence of events of interest is built by utilizing the following constraints.

- Temporal Constraint: The user specifies a *time interval of interest* with a start-time and an end-time, where only events occurring in this interval, e.g., from 2011-01-01 to 2012-12-31, are considered to analyze periodicity and to mine periodic patterns.

- Spatial Constraint: The user specifies a *region of interest* with a bounding box on a map or simply by giving names of instances (e.g., 'Germany' or 'Munich'). Only events occurring in this region are considered in mining periodic patterns.

- Conceptual Constraint: From the concept hierarchies for event contexts, the user selects some concepts that she is interested in. Then only events related to these concepts, i.e., events having paths to one of the concepts, are considered when mining for respective patterns.

In this chapter, we use $\mathcal{C}$ to denote the set of event constraints (temporal, spatial and conceptual constraints) specified by the user. Events that satisfy these constraints are used to build an event sequence for mining periodic patterns. We will detail this idea in Section 5.4.

As discussed before, event templates, generalized from events by using hierarchies, are key ingredients of periodic event patterns. However, a hierarchy for concepts of events is often large. Moreover, in some case patterns related to concepts that are too general might not be of interest to the user. Thus, we allow the user to specify

how far an event can be generalized by using a threshold $g_{max} \geq 0$ by employing the definition of context levels introduced in Section 3.6.2. More precisely, this threshold is used to constrain the specificity of periodic event patterns to be discovered. For example, if the user set the threshold g_{max} to 2, then the mining process will generate only contexts of Levels 0, 1, or 2 for event templates.

5.3.2 Event Co-occurrences

Aiming at periodic event patterns where a pattern consists of event topics that co-occur and repeat over time, this section describes how to determine which events in a given dataset co-occur. Basically, the user can specify how two events are related in space and time based on spatial and temporal relationships as introduced in Section 3.5 (Page 46). We detail this idea as follows.

- Spatial proximity: Given a spatial relationship $\mathcal{R}_s$ (Definition 3.28 - Page 47), two events e_1 and e_2 are said to be related in space iff $(e_1, e_2) \in \mathcal{R}_s$. For example, the user specifies an Euclidean distance threshold (e.g., 1 km) for locations to define $\mathcal{R}_s$.
- Temporal proximity: Given a temporal relationship $\mathcal{R}_t$ (Definition 3.28 - Page 50), two events e_1 and e_2 are said to be related in time iff $(e_1, e_2) \in \mathcal{R}_t$. Basically, the user specifies a time duration, for instance, 1 day, then two events are said to be related in time if they occur within a day.

Summing up, given a spatial relationship $\mathcal{R}_s$ and a temporal relationship $\mathcal{R}_t$, two events are said to be related in space and time (or they co-occur) if and only if they have both relationships $\mathcal{R}_s$ and $\mathcal{R}_t$.

5.4 Formulating Periodic Event Patterns

We now define *periodic event patterns*, or *p-patterns* for short, which describe regularities of event topics (specified by event templates) over time. Before giving a formal definition of p-patterns, we introduce the notations of *time slots*, *support segments*, and *event instance vectors*.

5.4.1 Time Slots

Given a time interval of interest (e.g., 3 years) and a temporal granularity chosen as a time unit (e.g., a day), the time interval is divided into *time slots* where the

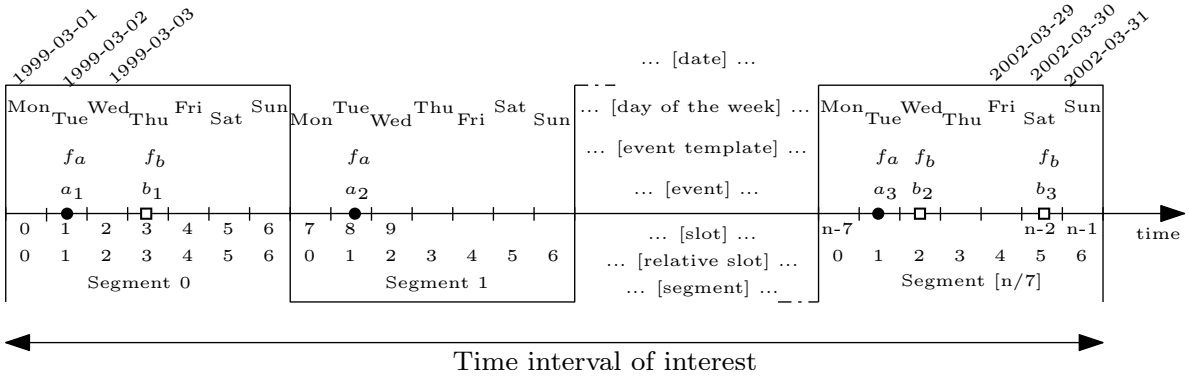


Figure 5.2: Illustration of time slots and segments. The time interval of interest is divided into slots (days). The first segment (the first week) consists of the first 7 slots. The second segment (the second week) consists of the next 7 slots, and so on. The event template f_a (whose instances are events a_1, a_2, a_3, etc.) is periodic but the event template f_b (whose instances are events b_1, b_2, b_3, etc.) is not.

duration of each slot is a time unit. For example, the time interval of about 3 years from 1999-03-01 to 2002-03-31 shown in Figure 5.2 consists of 1106 days, and each day corresponds to a time slot. We now give a formal definition of time slots.

Definition 5.1 (Time Slots) *Given a time interval T of interest, and a time unit t_u, the interval T is divided into consecutive **time slots**, or **slots**, for short, where the duration of each slot is t_u. The i-th slot of the interval T is called the time slot i.*

To formulate instances of patterns later on, we assume that the time unit is chosen for the input dataset of events such that each event is anchored at only one slot based on its occurrence time. For example, if the occurrence time of each event of a given dataset is specified at (or can be mapped to) the granularity day, then a day can be chosen as a time unit. The notation of time slots allows formulating *t-Segmentation* to model periodicities later on.

Definition 5.2 (t-Segmentation) *Let T be a time interval of interest, and n be the number of time slots of T. Given an integer $t > 0$, a **t-segmentation** is a segmentation of the time interval T such that the k-th segment ($k \in \{0, 1, .., \lfloor \frac{n}{t} \rfloor\}$), denoted $\mathcal{I}_k$, consists of slots from slot $(t * k)$ to slot $((t-1) + t * k)$.*

Given a set of event constraints $\mathcal{C}$, the set of events occurring in the i-th slot ($i \in \{0, 1, .., t-1\}$) of a segment $\mathcal{I}_k$ and satisfying all constraints in $\mathcal{C}$ is denoted as $\mathcal{I}_k[i]$.

Generally speaking, given a positive integer t and a time interval T, each segment of a t-Segmentation consists of t consecutive slots of the time interval T, where the

first segment consists of the first t slots, i.e., slots 0 to $(t-1)$; the second segment consists of the next t slots, i.e., slots t to $(2t-1)$; and so on. Note that if the number of slots n is not a multiple of t then some last slots will be truncated.

As an example, Figure 5.2 illustrates a 7-segmentation of 3 years from 1999-03-01 to 2002-03-31. In this figure, each segment consists of seven slots, where each slot corresponds to a day of the week (Monday, Tuesday,..., Sunday). Following this, the first segment represents the first week; the second segment represents the second week; and so on. The set of events found in the second slot of the first segment ($\mathcal{I}_0$), for instance, is $\mathcal{I}_0[1] = \{a_1\}$, while the set of events found in the fourth slot of the first segment is $\mathcal{I}_0[3] = \{b_1\}$ (assume that a_1 and b_1 satisfy all event constraints in $\mathcal{C}$).

With a given t-segmentation, an event template f is said to be *periodic* if there exists $i \in \{0, 1, .., t-1\}$ such that an instance of f is found in the i-th slot of *every segment (strictly periodic)* or in *most segments (relaxed periodicity)*. In Figure 5.2, the event template f_a is periodic because its instances (events a_1, a_2, and a_3) are found in every second slot (Tuesday) of the segments. However, the event template f_b is not periodic since its instances (b_1, b_2, and b_3) occur randomly.

Obviously, using such a strict form of periodicity like in the above example to formulate periodic patterns is not practical since real life patterns are often imperfect. Therefore, we take a relaxed form of periodicity into account in this chapter. For this, we introduce the notion of *support segments* in the next section.

In the following, we assume that a positive integer t is chosen for the segmentation of the time interval of interest T. That is, all the segments mentioned in the following section have the same length (duration), and each segment consists of t time slots.

5.4.2 Support Segments

In practice, instances of an event template f might be found in the i-th slot of most but not all segments. To relax the notion of periodicity, we define *support segments* and then introduce a *p-score* for an event template.

Definition 5.3 (ET Support) *Let f be an event template and $\mathcal{I}$ be a segment. The segment $\mathcal{I}$ **supports** the event template f at the i-th slot iff the condition $\exists e \in \mathcal{I}[i] : e \Vdash f$ holds.*

In other words, a segment supports an event template f at the i-th slot iff one can find an instance of the event template f in the i-th slot of the segment. For example, the first segment in Figure 5.2 supports an event template f_a at the second slot and

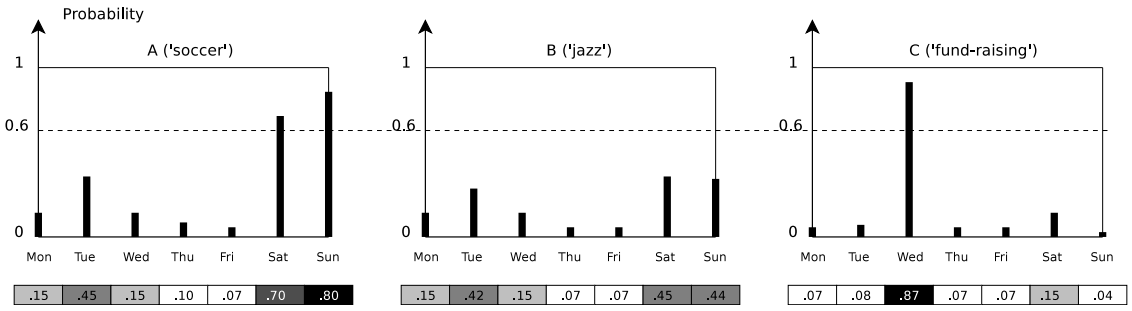


Figure 5.3: Schematic representations of the EIVs for three event templates A, B, and C, which are shown by using bar charts and color gradient. With a threshold $\delta = 0.6$, A and C are considered periodic, but B is not.

supports an event template f_b at the fourth slot because instances of f_a and f_b (a_1 and b_1) are found in the second and fourth slots, respectively, of this segment.

To characterize the periodicity of an event template with respect to the i-th slot, we compute a p-score as the ratio of the number of the support segments to the total number of segments as follows.

Definition 5.4 (p-score) *Let $\mathcal{S}_\mathcal{I}$ be the set of all segments based on a t-segmentation of a time interval T of interest. The p-score of an event template f with respect to the i-th slot is defined as*

$$\textit{p-score}(f, i) := \frac{|\{\mathcal{I} \in \mathcal{S}_\mathcal{I} \ : \ \mathcal{I} \textit{ supports } f \textit{ at the } i\textit{-th slot}\}|}{|\mathcal{S}_\mathcal{I}|} \qquad \in [0, 1]. \qquad (5.1)$$

In other words, the p-score(f,i) is simply the probability that instances of the event template f are found at the i-th slot of a segment. Since the function can be applied for any $i \in \{0, 1, ..., t-1\}$, we define a vector V, where $V[i]$=p-score(f,i), to represent the *periodic behavior* of an event template f. The vector V is called an *event instance vector (EIV)* of the event template f.

Definition 5.5 (Event Instance Vector) *Let $\mathcal{S}_\mathcal{I}$ be the set of all segments based on a t-segmentation of a time interval T of interest. An **event instance vector (EIV)** of an event template f is a vector of t components where the value of the i-th component is p-score(f, i).*

From a semantic point of view, one can analyze the periodicity of an event template based on its EIV. As an example, assume an event template A in the left side of Figure 5.3, whose instances are events related to '*soccer*', has an EIV [0.15, 0.45, 0.15, 0.1, 0.07, 0.7, 0.8]. This EIV shows that it is very likely to find events related

to '*soccer*' on Saturday and Sunday every week since the respective p-scores (0.7 and 0.8) are much higher than the p-scores of other slots.

Given a threshold $\delta \in [0, 1]$, an event template f is called to be *periodic* if there exists a component of the EIV of f such that the respective p-score is no less than δ. For example, the event templates A (whose instances are events related to *'soccer'*) and C (whose instances are events related to *'fund-raising'*) in Figure 5.3 are periodic with respect to a threshold $\delta = 0.6$, but the event template B (whose instances are events related to *'jazz'*) is not. We now introduce a definition of *periodic event templates*.

Definition 5.6 (Periodic ET) *Let f be an event template whose event instance vector is V, and $\delta \in [0, 1]$ be a predefined threshold. The event template f is called* ***periodic*** *iff there exists a component $V[i]$ of V such that $V[i] \geq \delta$.*

If there exists an event template f that is periodic with respect to a given threshold and a t-segmentation, then t is called a *period* for the event template f.

Since our aim is to find periodic patterns consisting of multiple event templates that co-occur, we will extend the definitions and notations above to model periodicities of a set of event templates in the next section.

5.4.3 Periodic Event Patterns

We now extend Definition 5.3 to the support of a set of event templates (ETs) and then formulate *periodic event patterns*.

Definition 5.7 (Pattern Support) *Let $F = \{f_1, f_2, ..., f_k\}$ $(f_i \prec f_j, i < j)$ be a set of event templates and $\mathcal{I}$ be a segment. The segment $\mathcal{I}$* ***supports*** *the set F at the i-th slot iff there exists an* ***instance*** *in the form of a tuple of k events $E = \langle e_1, e_2, ..., e_k \rangle \subseteq \mathcal{I}[i]$ such that the following conditions hold:*

1. $\forall j \in \{1, 2, ..., k\},\ e_j \in \mathcal{I}[i] \wedge e_j \Vdash f_j$,
2. $\forall e, e' \in E\ (e \neq e'),\ (e, e') \in \mathcal{R}_s \wedge (e, e') \in \mathcal{R}_t$, *where $\mathcal{R}_s$ and $\mathcal{R}_s$ are spatial and temporal relationships predefined by the user.*

With this definition, the p-score of a set of event templates is defined similarly to the p-score of a single event template (Definition 5.4). That is, given a set of segments $\mathcal{S}_\mathcal{I}$, the p-score of a set F of event templates is computed as

$$\text{p-score}(F, i) := \frac{|\{\mathcal{I} \in \mathcal{S}_\mathcal{I}\ :\ \mathcal{I} \text{ supports } F \text{ at the i-th slot}\}|}{|\mathcal{S}_\mathcal{I}|} \qquad \in [0, 1]. \qquad (5.2)$$

The vector V where $V[i]$=p-score(F,i) ($i = 0..t-1$) is then called an event instance vector (EIV) of F. Following this, we define *periodic event patterns*, or *p-patterns*, for short.

Definition 5.8 (P-Pattern) *Let F be a set of event templates and V be an event instance vector. A pair $\langle F, V\rangle$ is called a* ***periodic event pattern****, or* ***p-pattern****, for short.*

Given a threshold $\delta \in [0, 1]$, a p-pattern $P = \langle F, V\rangle$ is ***significant*** *iff the following condition holds: $\exists i \in \{0, 1, ..., t-1\},\ V[i] \geq \delta$.*

From the above definition, one can infer that if a p-pattern $P = \langle F, V\rangle$ is significant with respect to a given threshold δ, then a derived pattern $P' = \langle F', V'\rangle$, where F' is derived from F by replacing one or more elements (ETs) by their generalizations, is also significant with respect to δ. This fact is stated in the following lemma.

Lemma 5.1 *Let $P = \langle\{f_1, f_2, ..., f_k\}, V\rangle$ and $P' = \langle\{f'_1, f'_2, ..., f'_k\}, V'\rangle$ be two p-patterns such that $\forall i \in \{1, 2, ..., k\},\ (f'_i = f_i) \vee (f'_i \in f_i^{\Uparrow})$. If the p-pattern P is significant with respect to a given threshold δ, then P' is also significant with respect to δ.*

Proof: Since every ET $f' \in F'$ is a generalization of a respective ET $f \in F$, every segment supporting F at the i-slot also supports F' at the i-th slot. Therefore, p-score(F',i) $\geq$ p-score(F,i) ($\forall i \in \{0, 1, ..., t-1\}$).
On the other hand, since F is significant with respect to δ, there exists an integer $i \in \{0, 1, ..., t-1\}$ such that p-score(F,i) $\geq \delta$. Thus, p-score(F',i) $\geq$ p-score(F,i) $\geq \delta$. Hence, F' is significant with respect to δ. □

Lemma 5.1 motivates us to find only the *most specialized p-patterns* that are significant. From such a compact set of results, it is trivial to generate the full set of all significant p-patterns by using the generalization operators ($\uparrow$, $\Uparrow$) on ETs. In summary, we describe the problem of mining p-patterns as follows:

Given:

- a dataset $\mathcal{D}$ consisting of event instances of the form $\langle e_{id}$, Context, Time, Location$\rangle$,
- a set of hierarchies $\mathcal{H}$ for context, time, and location components,
- a time unit t_u and an interval of interest T,

- a set of event constraints $\mathcal{C}$ (temporal, spatial and conceptual constraints),
- a generalization threshold g_{max} for event templates,
- a spatial relationship $\mathcal{R}_s$ and a temporal relationship $\mathcal{R}_t$ for event co-occurrences,
- a p-score threshold δ.

Find: all *most specialized p-patterns* that are significant with respect to the threshold δ, the relationships $\mathcal{R}_s$, $\mathcal{R}_t$, and the event constraints in $\mathcal{C}$ such that the level of each event context in the patterns is less than or equal to g_{max}.

We describe our approach to the discovery of all p-patterns that meet the above requirements in the following section.

5.5 Mining Periodic Event Patterns

Our approach to the discovery of significant p-patterns from an event dataset consists of two stages. In the first stage, all possible periods are identified for mining p-patterns by analyzing the occurrences of events. For each period detected in the first stage, significant p-patterns with respect to this period are then discovered in the second stage. The two stages are detailed in the following sections.

5.5.1 Detecting Periods

Although the user can specify a period t corresponding to the kinds of periodicities of p-patterns (e.g., weekly or monthly) that she is interested in, we also include a step for automatically detecting periods from a dataset of events to make our framework more general. This task can be executed independently and helps the user to select meaningful periods for finding p-patterns. For this, we propose an approach to detecting all possible periods based on a lemma described below.

Lemma 5.2 *Let $P = \langle F, V \rangle$, and $P' = \langle F', V' \rangle$ be two p-patterns. If P is significant with respect to a given threshold δ and $F' \subset F$, then P' is also significant with respect to the threshold δ.*

Proof: Since $F' \subset F$, for each segment $\mathcal{I}$ supporting F at the i-th slot, it is trivial to show that $\mathcal{I}$ also supports F' at the i-th slot. Hence, the set of segments supporting F is a subset of the set of segments supporting F', at the i-th slot. From Equation 5.2,

we then have p-score(F',i) $\geq$ p-score(F,i) ($\forall i \in \{0, 1, ..., t-1\}$).
On the other hand, since P is significant with respect to δ, there exists an integer $i \in \{0, 1, ..., t-1\}$ such that p-score(F,i)$\geq$ δ. So, p-score(F',i) $\geq$ p-score(F,i)$\geq$ δ. Therefore, F is significant with respect to δ. □

Lemma 5.2 states that the Apriori property can be exploited to find all significant p-patterns. Accordingly, if one can find a significant p-pattern $P = \langle F, V \rangle$ with respect to a threshold δ, then all size-1 p-patterns where each p-pattern consists of a single ET $f \in F$ are also significant with respect to the threshold δ. Therefore, one can identify all possible, meaningful periods for p-patterns by separately considering periodicities of (single) ETs generated from events.

A straightforward method to detect periods for an ET f is to construct a binary sequence S from its instances (events), and then identify periods from that sequence. The binary sequence S is a sequence of n elements (n is the number of time slots), where $S[i] = 1$ ($0 \leq i \leq n-1$) if the time slot i contains an event that is an instance of f, and $S[i] = 0$ otherwise. The binary sequence S is processed further to detect periods by using any periodicity analysis method mentioned in Section 5.2.

However, considering all generated ETs for detecting periods might be time consuming since the number of ETs is often large. However, by applying Lemma 5.1 for a special case where both the p-patterns P and P' are of size-1, one can infer that if an ET f is periodic with respect to a threshold δ, then every ET $f' \in f^{\Uparrow}$ is also periodic with respect to δ. This consequence allows one to obtain all possible periods of an ET by analyzing the periodicity of its generalizations. Hence, instead of considering all possible ETs that can be generated from events, we detect periods only for *most generalized ETs* (according to the threshold g_{max}).

Through the above procedure, we finally obtain a set of all possible periods and then the user can use all of these periods or just select the periods that she is interested in for the next stage.

5.5.2 Finding P-Patterns

Since each period is considered separately, this section describes a process to mine p-patterns under the assumption that the current period is t. Based on Lemmas 5.1 and 5.2, we propose an approach to the discovery of significant p-patterns from an event dataset.

Basically, our approach consists of two steps. The first step is to generate candidates of p-patterns in the order of the size of candidates (number of ETs), that is,

size-1 candidates (consisting of one ET) are first generated from events; following that, size-2 candidates are generated by using the analysis of the size-1 candidates; and so on. At each size, the next step is performed to generate generalizations of candidates in the order of most specialized to most generalized ones. For each candidate, its EIV is computed and tested with the threshold δ for termination of the procedure.

For the first step and second step, we propose Algorithms 5.1 and 5.2, respectively. They are described in detail in the following.

Algorithm 5.1: P-Pattern-Miner

Input:
(a) $\mathcal{D}$: event dataset
(b) $\mathcal{H}$: set of hierarchies
(c) t: time period
(d) δ: threshold for periodic score
(e) $\mathcal{C}$: set of constraints
(f) $\mathcal{R}_s, \mathcal{R}_t$: spatial, temporal relationships
(g) g_{max}: maximum level of generalization for event templates
Output: Set of most specialized p-patterns

1 $\mathcal{D}_c = \{\text{e} \in \mathcal{D} : \text{e satisfies } \mathcal{C}\}$; `/* only consider events satisfying constraints */`
2 $L_1 = \{\}$;
3 **foreach** Event $e \in \mathcal{D}_c$ **do**
4 **foreach** Event Template $f \in e^{\uparrow}$ **do**
5 Create a size-1 candidate P_1 from f and t;
6 $L_1 = L_1 \bigcup$ Search_PPatterns(P_1, $\mathcal{D}_c$, $\mathcal{H}$, δ, g_{max});
7 $result = L_1$; k=1;
8 **while** $L_k \neq \{\}$ **do**
9 $L_{k+1} = \{\}$;
10 **foreach** $P_k = \langle F, V \rangle \in L_k$ **do**
11 **foreach** $P_1 = \langle \{g\}, V' \rangle \in L_1$ **do**
12 **if** g is not a generalization of any $f \in F$ **then**
13 Create size-(k+1) candidate P_{k+1} from $F \cup \{g\}$ and t;
14 $L_{k+1} = L_{k+1} \bigcup$ Search_PPatterns(P_{k+1}, $\mathcal{D}_c$, $\mathcal{H}$, δ, g_{max});
15 $result = result \bigcup L_{k+1}$;
16 $k = k + 1$;
17 **return** $result$;

Algorithm 5.1, named P-Pattern-Miner, is the main procedure to find significant p-patterns for a given period t. First, a subset $\mathcal{D}_c$ of events that satisfy the constraints in $\mathcal{C}$ is created from the dataset $\mathcal{D}$ (Line 1). The rest of P-Pattern-Miner consists of two phases: (1) to find size-1 p-patterns that are significant, and (2) to generate and

test to find size-k ($k \geq 2$) p-patterns (consisting of k ETs that are to be checked for co-occurrence).

Lines 2-6 of Algorithm 5.1 show the process to discover size-1 significant p-patterns (the first phase). For each event in $\mathcal{D}_c$, ETs are generated by employing the (direct) generalization operator $\uparrow$. For each generated ET, a size-1 candidate is created with an EIV initialized to zero. Most generalized, significant p-patterns derived from such a candidate are discovered by a function shown in Algorithm 5.2, called *Search_PPatterns*. This function is detailed as follows.

Algorithm 5.2: Search_PPatterns(P, $\mathcal{D}_c$, $\mathcal{H}$, δ, g_{max})

```
   /* Breadth-first Search from P to find p-patterns satisfying threshold δ   */
 1 result={}; Queue=[P];
 2 while Queue is not empty do
 3 |   q=Queue.dequeue();
 4 |   Compute the EIV V of q from D_c;
 5 |   if max_{i=0}^{t-1}{V[i]} ≥ δ then
 6 |   |   result = result ⋃ {q};
 7 |   else
 8 |   |   Compute the set q^↑ from q and H;
 9 |   |   foreach Pattern r ∈ q^↑ do
10 |   |   |   if level(r) ≤ g_max then
11 |   |   |   |   Queue.enqueue(r);
12 return result;
```

Generally, the function *Search_PPatterns* uses a Breadth-First-Search to find the *most specialized generalizations* of a candidate P that are significant with respect to the threshold δ. As shown in Algorithm 5.2, a queue is employed to store candidates generalized from a given candidate P. In Lines 3-4, the EIV of each candidate in the queue is sequentially computed to determine how significant the current candidate is. If a candidate is significant, it is stored in the result set (Lines 5-6). Otherwise its direct generalizations satisfying the threshold g_{max} will be added to the queue for further iterations (Lines 8-11). This process stops when the queue of candidates is empty.

We now return to Algorithm 5.1. The rest of Algorithm 5.1 (Lines 7 to 16) describes the second phase containing a while-loop to find p-patterns of size-2, size-3, etc. At an iteration step k ($k \geq 1$), size-(k+1) p-patterns are discovered by using a size-k p-pattern P_k and a size-1 p-pattern P_1. To ensure the size of new p-patterns is $(k+1)$, each ET in P_k must not be a generalization of the ET in P_1 (Line 12).

Then a size-(k+1) candidate (P_{k+1}) is created and passed to the *Search_PPatterns* function to find size-(k+1) p-patterns (Lines 13 and 14). Finally, a set of p-patterns of all sizes is returned.

The runtime of our approach depends on the number of generated candidates, and this number heavily depends on the characteristics of the input dataset and given hierarchies. Estimating the number in the worst-case is meaningless since all possible candidates might be a very large number but the real number in practice is often much smaller. Therefore, similar to other pattern mining approaches that employ the apriori method, we only analyze the experimental runtime for real datasets, as shown in the next section.

5.6 Experimental Evaluation

We demonstrate the utility of our approach using events, e.g., festivals, concerts, sports, etc., crawled from the Website *eventful.com* from 2009 to 2013. Such events, stored as RDF facts, are transformed to the form $\langle e_{id}, Context, Time, Location \rangle$ for the purpose of utilizing the event model introduced in Chapter 3. The objective of our evaluation is to show how to effectively apply the framework to mine p-patterns from RDF data and to demonstrate that interesting periodic event patterns can be discovered from such a dataset. In the following, we describe the setup of our experiments and present some interesting patterns extracted from different datasets. Our framework is implemented in Java and runs with 24GB heap size. All experiments were performed on an Intel Xeon 2.27GHz with 48GB RAM, running Ubuntu 64bit.

5.6.1 Datasets and Experimental Setup

In this section, we describe the data source, named EVF, used in our experiments as well as the pre-processing steps to make such datasets available for our mining framework.

As mentioned in Section 4.7.2, each event in the EVF source consists of multiple attributes. Similar to the experiments described in the previous chapter, the context, time, and location components of an event are mapped to the following attributes: the event identifier, start-time, and venue identifier, respectively. Note that event identifiers are used for two purposes, i.e., to distinguish an event from others and to attach event contexts to a concept hierarchy built from event tags. From the venue

Table 5.1: Top countries consisting of most events in the EVF dataset. The column *'All'* corresponds to the number of events related to a specific topic (*'sports', 'festival', and 'religion'*). The column *'Recur.'* corresponds to the number of only recurring events related to that specific topic.

Country	**Total Events for all topics**	**Topic 'sports'**		**Topic 'festival'**		**Topic 'religion'**	
		All	Recur.	All	Recur.	All	Recur.
United States	5,160,952	395,021	69,850	94,567	16,489	70,483	7,189
United Kingdom	419,641	14,645	447	12,125	987	3,627	234
Canada	230,219	8,968	1,427	5,716	718	3,372	555
Germany	189,395	3,337	88	3,088	244	100	27
Australia	111,605	3,061	217	4,502	371	976	105
France	68,883	5,567	196	5,612	290	38	0
India	38,049	1,618	45	2,967	122	226	97
Italy	37,704	2,324	17	1,047	99	45	4
Ireland	33,025	585	25	1,734	152	89	16
Switzerland	32,749	424	2	967	46	20	2

identifier, one can retrieve geographic attributes of the corresponding location for checking spatial constraints later on.

To determine the time interval T of interest, we counted the events in each month. Based on the result (see Figure 4.11), we considered events only in three years, 2010, 2011, and 2012, for our experiments. The number of events in this time interval is about 7 million.

Among the attributes of events, the attribute *"recur-string"* describes that the corresponding event is either a *singleton* or *recurring* event. The values for this attribute might be empty (for singleton events) or a string describing a periodicity (recurring events), such as *"monthly on the 1st Friday for 3 times"*, or *"weekly on Mondays and Wednesdays until December 31, 2011"*. Although we are interested in both types of events, this attribute helps us to determine which regions and event topics (tags) are potentially useful to discover periodic event patterns. Before detailing this idea, we describe a concept hierarchy for event tags in the following.

To employ concept hierarchies and conceptual constraints later on, we utilize tags associated with events. A concept hierarchy for tags was built using the same method as described in Section 4.7.2. Basically, a *sub-class-of* relationship between two tags is specified by using the following assumption: a tag T_1 is a subclass of a tag T_2 if the set of events having the tag T_1 is a subset of the set of events having the tag T_2. Through this process, we obtained a hierarchy consisting of about 5000 tags of 5 levels, where the last three levels contain only few tags (about 2%), and most of them do not produce interesting patterns, such as *'photo', 'program', or 'member'*. Thus, we set the threshold g_{max} to 2 for this hierarchy.

We now use the concept hierarchy described above to select events based on topics (tags). Table 5.1 shows the numbers of events occurring in several countries for all topics and for some particular topics that are prevalent and common (in this case, *'sports', 'festival, and 'religion'*) in the EVF data source. This table also shows the number of recurring events for each pair of a country and a topic. Since the number of (recurring) events in the United States is the largest, we will consider events in this area to create datasets for our experiments later on.

In the next sections, we first analyze periodicities existing in the EVF data and then show typical p-patterns of the periods obtained from the periodicity analysis.

5.6.2 Periodicity Analysis

As discussed before, we selected the time interval of interest of three years, 2010, 2011, and 2012. For the time unit t_u, we select a day as duration for slots. We now employ the method described in Section 5.5.1 to detect meaningful periods for the discovery of p-patterns.

For each pair of a region (country) and an event topic (tag), we generate a binary sequence, where the i-th element is either 0 or 1 depending on whether or not an event related to that topic occurs at the i-th slot of the time interval of interest. Such a sequence is then analyzed to detect periods with any state-of-the-art method (see Section 2.2.3). In our experiments, we tried the following methods: (1) computing raw periodogram using the Fast Fourier Transform [113], (2) computing the autocorrelation [114], and (3) using the Lomb-Scargle algorithm [74]. For all these methods, significant periods are easily detected based on the peaks. For example, Figure 5.4 shows the result for the topic *'fleamarket'*, where one can see that the period 7 days is much more significant than other periods. For some event topics that are very common, such as *'music'*, *'training'*, or *'fitness'* (shown in Figure 5.5), no significant period is detected.

Through this process, we obtained different types of periods, such as 3 days, 5 days, or 10 days. However, the periods in most cases are either 7 days (weekly) or 28-30 days (monthly).

Selecting a period obtained from this process as an input, we then find significant p-patterns for that period. We detail this in the following section.

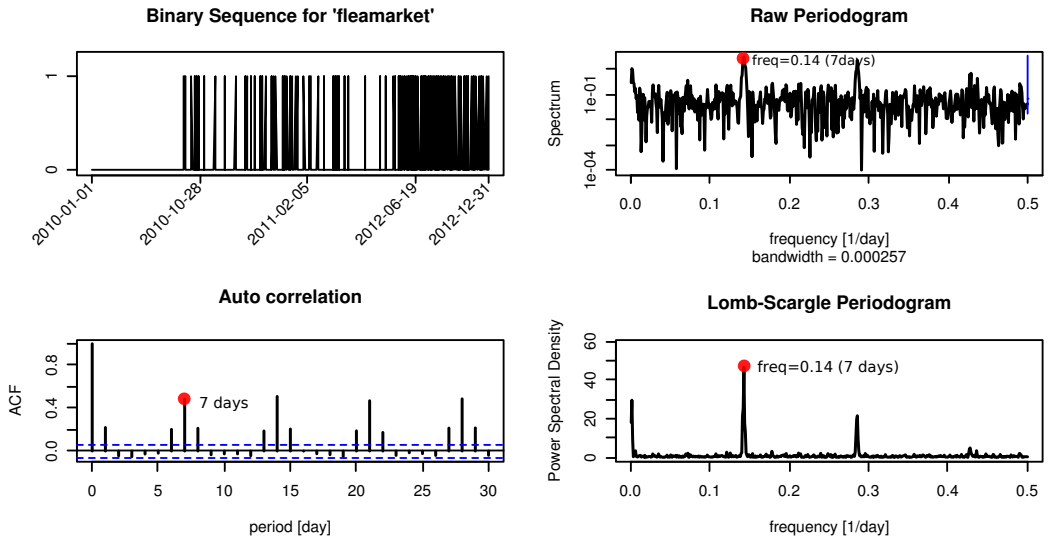


Figure 5.4: Periodicity analysis for the topic *'fleamarket'* in the US. The most likely period is 7 (days).

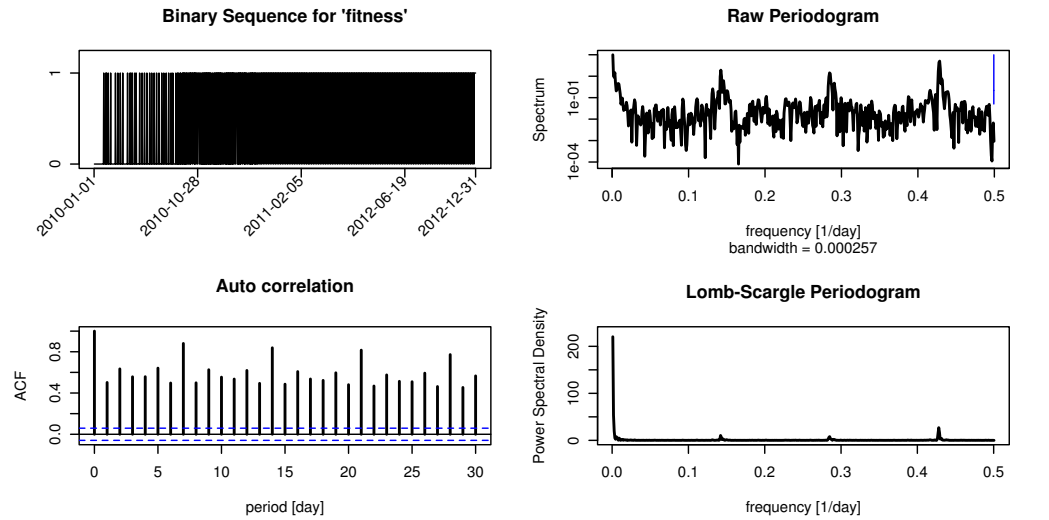


Figure 5.5: Periodicity analysis for the topic *'fitness'* in the US. No significant period is detected for this topic.

Table 5.2: List of exprimental datasets. S.Events represent singleton events, where each event consists of one instance. R.Events represent recurring events, where each event consists of multiple instances. An E.Instance is an instance of either a singleton or a recurring event.

Dataset	**Events related to**	**S.Events**	**R.Events**	**E.Instances**
Religion	'religion'	63,294	7,189	102,131
Festival	'festival'	78,078	16,489	107,399
Clubs & Associations	'clubs & associations'	306,865	5,471	352,922
Sports	'sports'	325,171	69,850	431,045
Community	'community'	425,451	98,517	563,755

5.6.3 P-Pattern Discovery

For the purpose of interpreting the results of p-patterns later on, we only consider weekly and monthly patterns. For weekly patterns, we select a day as a time unit for slots. For monthly patterns, a month is divided into 3 sections of around 10 days (early: 1-10, middle: 11-20 and late: 21-end), and each such section is considered a time unit.

Table 5.2 shows a list of the datasets created for our experiments. One can see that the size of a dataset is determined by the total instances of events, where each event is either a singleton (consists of one instance) or recurring (consists of multiple instances). Note that the time interval of interest for both datasets is from 2010 to 2013, and all selected events occurred in the US.

To specify the spatial and temporal relationships ($\mathcal{R}_s$ and $\mathcal{R}_t$), we simply assume that two events are said to be related in space and time if they occur at the same address and in the same time slot.

Event templates describing topics of events are generalized by using the concept hierarchy described in Section 5.6.1. The time and location hierarchies are specified as $(Day \rightarrow Month \rightarrow Year \rightarrow All_{time})$ and $(Address \rightarrow City \rightarrow State \rightarrow All_{loc})$. We omit country in the location hierarchy since all the events are in the same country, here the US.

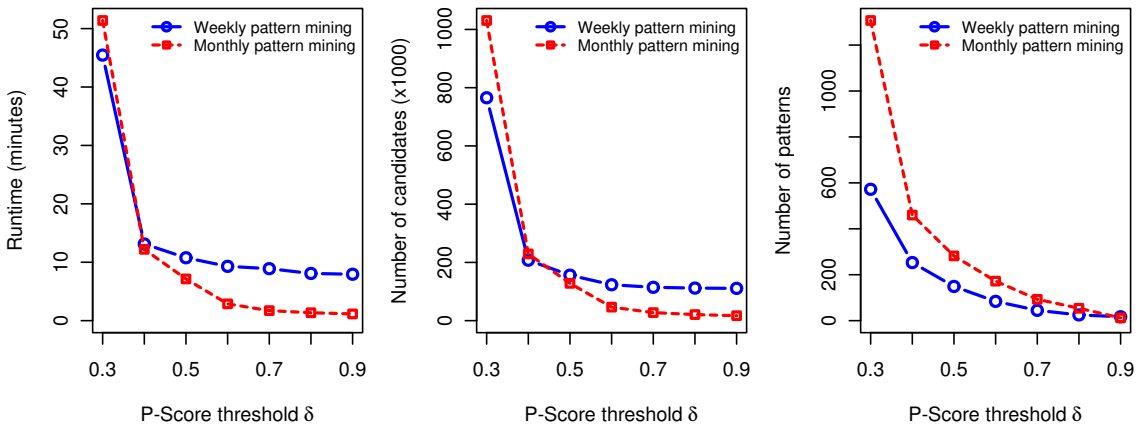


Figure 5.6: The runtime, number of candidates, and number of discovered p-patterns for the dataset 'Religion', at different p-score thresholds.

For each dataset, we ran the algorithm presented in Section 5.5 with different settings for the threshold δ, ranging from 0 to 1. Obviously, the smaller the value, the more p-patterns are found. However, a large number of candidates might be

Table 5.3: Statistics of experimental results ($\delta = 0.5$).

Dataset	Event Instances	Period	Time Slots	Candidates	Patterns	Runtime (Minutes)
Religion	102,131	Weekly	1,105	156,180	149	13.8
		Monthly	108	172,243	376	7.4
Festival	107,399	Weekly	1,105	145,592	239	13.8
		Monthly	108	408,231	694	22.1
Clubs & Associations	352,922	Weekly	1,105	167,719	188	38.7
		Monthly	108	254,846	501	35.4
Sports	431,045	Weekly	1,105	1,670,806	1,842	584.7
		Monthly	108	655,860	1,565	189.9
Community	563,755	Weekly	1,105	1,922,107	2,102	621.1
		Monthly	108	877,110	1,965	233.9

generated for a small value for δ. As a result, the runtime will be large. In our experiments, more than 20 millions candidates are generated in more than 50 hours when $\delta \leq 0.2$. As an illustration, one can see in Figure 5.6 that decreasing the value for δ causes increasing the number of candidates, the number of patterns, and thus, the runtime for mining the dataset 'Religion'.

Table 5.3 shows the number of candidates, the number of patterns, and execution times for $\delta = 0.5$ and for each dataset listed in Table 5.2. One can see that the numbers of candidates generated from the two datasets 'Sports' and 'Community' are much larger than from the others, and thus, the runtimes for these datasets are also much larger than the runtimes for the others.

Since the interestingness of a p-pattern is subjective, it is not trivial to rank the discovered p-patterns and select the top most "interesting" ones. Here we manually selected p-patterns to present by considering the distributions of their p-scores and event topics. Table 5.4 shows typical weekly patterns found using the threshold $\delta = 0.5$, that is, a p-pattern is significant if there are more than 50% of the segments (weeks) supporting it. Since our datasets consists of events in only 3 years (2010, 2011, and 2012), to satisfy that threshold, all time components are obviously generalized to All_{time}. Thus, we present event templates of p-patterns in the form of *subject.location*.

As shown in Table 5.4, sports events related to '*boxing*' are likely to be found on Fridays whereas sports events related to '*walking*' or '*cycling*' are likely to be found on Saturdays and Sundays. Besides, some p-patterns consisting of two or more event templates describe co-occurrences. For example, we found that co-occurrences of an event related to '*kid*' and another event related to '*church*' most likely appear on Sundays, as shown in the results of the dataset 'Religion'. Moreover, some patterns are only valid at a specific location, for instance, a pattern describing concerts on Fridays

Table 5.4: Selected patterns with their p-scores extracted from the experimental datasets. Time components are omitted; '*' denotes "for all locations". P-scores satisfying the threshold $\delta = 0.5$ are shown in boldface.

Event templates	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Some patterns extracted from the dataset 'Sports'							
{boxing.*}	0.02	0.01	0.04	0.08	**0.51**	0.38	0.08
{walking.*}	0.34	0.04	0.09	0.15	0.17	**0.77**	**0.72**
{cycling.*}	0.23	0.44	0.37	0.41	0.41	**0.92**	**0.66**
{fundrais.Arizona}	0.10	0.09	0.06	0.11	0.18	**0.53**	0.15
{marathon.*}	0.40	0.36	0.30	0.37	**0.53**	**0.75**	**0.74**
{education.*; outdoor & recreation.*; food.*}	0.08	0.30	0.36	**0.56**	**0.54**	**0.70**	0.47
{fishing.*; outdoor & recreation.*}	0.42	**0.54**	0.41	0.41	0.30	**0.68**	0.47
Some patterns extracted from the dataset 'Religion'							
{meditation.Atlanta,Georgia}	0.07	0.04	0.18	0.05	0.07	**0.51**	0.09
{taichi.*}	0.23	**0.55**	0.01	0.06	0.03	0.27	0.22
{church.Houston,Texas}	0.44	0.14	0.42	0.29	0.21	0.19	**0.57**
{pray.*}	**0.65**	**0.61**	**0.65**	**0.65**	**0.63**	**0.66**	**0.97**
{yoga.*; support.*}	0.11	0.32	0.03	**0.56**	0.03	0.26	0.28
{kid.*; church.*}	0.00	0.00	0.01	0.00	0.01	0.02	**0.53**
Some patterns extracted from the dataset 'Community'							
{kid.Atlanta,Georgia}	**0.65**	**0.68**	**0.77**	**0.57**	0.41	**0.77**	0.17
{pubtrivia.*}	0.00	0.02	0.34	**0.65**	0.31	0.00	0.01
{concert.Milwaukee,Wisconsin}	0.11	0.18	0.23	0.35	**0.75**	0.39	0.32
{visitomaha.Omaha,Nebraska}	0.22	0.25	0.19	**0.51**	**0.58**	**0.72**	**0.54**
{learning.*; family.*}	**0.58**	0.34	0.44	0.22	0.19	0.24	0.19
{drink.*; outdoor & recreation.*}	0.19	0.03	0.05	0.03	0.21	**0.51**	0.10
Some patterns extracted from the dataset 'Festival'							
{fleamarketsal.California}	0.02	0.36	0.03	0.03	0.06	0.34	**0.71**
{food.Missouri}	0.03	0.03	0.04	0.04	0.18	**0.51**	0.09
{music.Arizona}	0.04	0.08	0.16	0.23	0.45	**0.54**	0.34
{beer.*}	0.09	0.11	0.12	0.24	0.43	**0.59**	0.24
{familyfun.*}	0.09	0.09	0.28	0.27	0.43	**0.53**	0.34
Some patterns extracted from the dataset 'Clubs & associations'							
{metaphysical.The Crystal Closet,Sanford,Florida}	0.00	0.00	0.34	0.00	0.00	**0.54**	0.01
{fundrais.Texas}	0.11	0.22	0.18	**0.51**	0.19	0.46	0.45
{business.Washington}	0.15	0.32	0.42	**0.62**	0.25	0.14	0.10
{jazz.California}	0.02	0.37	0.46	0.21	**0.53**	0.16	0.10
{technology.*}	**0.64**	**0.58**	**0.59**	**0.56**	0.39	0.39	0.20
{meeting.*}	**0.53**	0.34	0.39	0.44	0.29	0.49	0.16
{basketball.*}	0.08	0.07	0.11	0.17	0.07	0.14	**0.53**
{conferenc.*; club & association.*}	0.37	0.44	**0.54**	0.34	0.19	0.37	0.33

in Milwaukee - Wisconsin (from the dataset 'Community') or a pattern describing flea markets on Sundays in California (from the dataset 'Festival'). In addition, several p-patterns describe that event topics frequently appear most days of the week, except for one or two days. For example, events related to '*technology*' or '*meeting*' usually occur on most days of the week but not on Sundays, as shown in the results of the dataset 'Clubs & associations'.

Also monthly patterns were found in our experiments. Table 5.5 shows some typical p-patterns extracted from the five datasets. From this table, one can see several patterns describing co-occurrences of event topics, for instance, a co-occurrence of an event related to '*fundrais*' (fundraising) and another event related to '*fishing*' mostly appears early each month. Interestingly, a p-pattern whose p-scores at all slots satisfying the threshold δ can be interpreted as a correlation between an event topic and

Table 5.5: Selected patterns with their p-scores extracted from the experimental datasets. Time components are omitted; '*' denotes "for all locations". P-scores satisfying the threshold $\delta = 0.5$ are shown in boldface.

Event templates	Early	Mid	Late
Some patterns extracted from the dataset 'Sports'			
{kid.Georgia}	**0.53**	0.36	0.08
{business.*; sport.*}	0.28	**0.69**	0.42
{fundrais.*; fishing.*}	**0.50**	0.06	0.03
{beerpong.California; outdoor & recreation.California}	**0.53**	0.00	0.00
{trips.*; bik.*; fundrais.*}	**0.61**	0.22	0.22
{trips.*; fishing.*; fundrais.*; outdoor & recreation.*; sal.*}	**0.58**	0.22	0.22
Some patterns extracted from the dataset 'Religion'			
{spirituality.Florida}	0.33	0.39	**0.75**
{singing.*}	**0.61**	0.42	0.25
{party.*; religion & spirituality.*}	0.25	0.36	**0.64**
{art.*; music.*}	0.19	0.14	**0.58**
{spiritual.*; religion & spirituality.*; learning.*}	0.17	**0.53**	0.11
Some patterns extracted from the dataset 'Community'			
{art.Missouri}	**0.83**	0.47	0.28
{cooking.*}	**0.56**	**0.92**	**0.61**
{vendor.*}	**0.72**	**0.53**	0.33
{family.*; music.*}	**0.81**	**0.53**	0.39
{performing & art.*; local.*}	0.25	**0.72**	**0.56**
{club.*; local.*; art.*}	**0.75**	0.39	**0.72**
Some patterns extracted from the dataset 'Festival'			
{livemusic.Florida}	**0.50**	0.36	0.22
{filmfestival.,Los Angeles,California}	**0.61**	**0.56**	**0.61**
{concert.*; art.*}	**0.56**	0.31	0.19
{food.Florida; outdoor & recreation.Florida}	**0.53**	0.31	0.08
{comedy.*; music.*; festival & parad.*}	0.19	**0.56**	**0.53**
{singl & social.*; concert.*; festival.*; performing & art.*}	0.11	0.14	**0.50**
Some patterns extracted from the dataset 'Clubs & associations'			
{donation.*}	**0.53**	0.14	0.28
{technology.Michigan}	0.03	**0.78**	0.17
{animal.Maryland}	0.08	**0.50**	0.11
{politics.Texas}	0.28	**0.61**	0.19
{smallbusiness.*}	0.14	0.42	**0.50**
{book.*; club & association.*}	0.33	**0.67**	**0.75**
{conferenc.*; club & association.*; singl & social.*}	0.19	**0.64**	0.25

a location. For example, as shown in the results of the dataset 'Festival' in Table 5.5, one can infer that film festivals and Los Angeles have a relationship because it is very likely (high probability) that an event related to film festivals is found any time at that location.

5.7 Discussion

In this chapter, we presented a framework to mine periodic patterns from datasets of events in the presence of conceptual, temporal, and spatial hierarchies. Such patterns represent topics of events that co-occur and repeat over time. Event topics are modeled based on event templates that were introduced in Chapter 3. The experimental results show that interesting p-patterns can be effectively discovered based on

the proposed approach. These patterns can be further exploited to predict upcoming events, for service and product suggestions, or to detect outliers in a dataset.

From the experiments, one can see some patterns whose p-scores at slots are all high. Although less information about periodicities are gained from such patterns, they can be used to extract correlations between an event topic and a particular location, e.g., as seen in the last example of film festivals and Los Angeles in the previous section. Furthermore, this kind of information can be exploited to annotate locations. For example, a tag '*film festival*' can be attached to the location Los Angeles if one knows the pattern describing that film festivals usually occur in Los Angeles. However, all patterns that consist of very common topics, such as '*music*' or '*sport*' might have the same characteristic, i.e., the p-score of each slot is high. For the purpose of extracting semantic annotations for locations, we propose another approach, as described in the next chapter.

Chapter 6

Extracting Semantic Annotations for Locations from Event Data

In Chapters 4 and 5, we exploited correlations of events for interesting patterns. Here we demonstrate that correlations of event components can be utilized for valuable knowledge as well. In particular, we focus on the problem of extracting semantic annotations for locations from event data. Such information about locations provides a semantically rich basis for location search, topic-based location clustering or recommendation services. However, little work has been done yet to extract such correlations from event datasets to annotate locations.

By employing the event model introduced in Chapter 3, in this chapter, we present our approach to the discovery of semantic annotations for locations from event data. We demonstrate the utility of extracted annotations in hierarchical clustering for locations, where the similarity between two locations is defined on the basis of their common event topics. Additionally, taxonomies of locations are then built from the analysis of location clusters. To deal with periodic updates of event datasets, we furthermore give a scalable and efficient approach to incrementally update location annotations. To demonstrate the performance of our approach, we use real event datasets crawled from the Website *eventful.com*.

In a paper by Le et al. [67], we presented some initial ideas of extracting semantic annotations for locations from event data. In this chapter, we provide more elaborate explanations for the notations, definitions, algorithms, and experimental results. We also discuss how to employ the framework in real applications.

6.1 Introduction

The main difference between a '*place*' and a position is that a place is represented as a human-readable description of a geographic location rather than just a geographic coordinate. Such descriptive information about locations is essential for location-based services (LBS), for instance, location recommendation, location-based mobile advertising, or social event recommendation [29, 47, 93]. Typically, a data source managing information about locations provides various attributes of a location for an LBS application, including the name, address, description, comments of users, and metadata such as tags. From a semantic perspective, especially attributes such as a description or tags associated with a location are useful in semantic location search, e.g., "*Find all bars that feature live music on weekends*".

Unfortunately, such descriptive attributes detailing location information tend to be poor in many data sources. For example, our analysis shows that there are about one million locations in a dataset of events crawled for the years 2011 and 2012 from the Website *eventful.com*, but only about 10% of them contain descriptions or tags. Moreover, querying based on simple text matching of descriptions or tags cannot take into account concept hierarchies that might exist for locations, time, or event topics. For example, using suitable concept hierarchies, the description "*live jazz on Saturdays*" may be considered a matching for the query "*live music on weekends*". Therefore, enriching information about events at different levels of granularity and abstraction is necessary and useful.

Several methods have been proposed to extract semantic annotations for locations. However, some of them heavily depend on the location data provided by external sources such as Wikipedia or the Google Maps API [19]. Other approaches exploit either user-tags of Flickr data, e.g., [94, 95], or the check-in data of users from online social networks, e.g., [47, 128]. Such types of user generated content are often sparse, noisy and sometimes even inaccurate. We will discuss these approaches in detail in Section 6.2.

On the other hand, numerous data sources managing information about events are available on the Internet. This includes popular Websites such as *last.fm*, *eventim.de*, or *eventful.com*. Although in these sources the event data are less noisy (and more accurate) than in other georeferenced social media, there are still challenges in fully exploiting such information, as concept hierarchies, either explicitly or implicitly, exist for event topics, locations, and time.

In this chapter, we aim at extracting semantic annotations for locations from event data by exploiting correlations among geographic locations, time and event topics. Intuitively, some events occur more likely at some place/time than at other places/times. For example, events related to the topics '*live music*', '*dance*' or '*party*' likely occur at a bar or club at weekends, whereas events related to '*conference*' or '*talk*' likely occur at a university on working days. Identifying and exploiting such correlations from event data is important to enrich data sources of locations with meaningful descriptions. However, to the best of our knowledge, such an approach has not been developed yet.

To determine correlations as the ones mentioned above, we distinguish three kinds of event topics (ETs) with respect to a given location: (1) ETs that are prevalent at that location but uncommon at other locations, (2) ETs that are popular at all locations, and (3) ETs that rarely occur at that location but commonly occur at other locations. Clearly, the ETs of the first category are more important and descriptive to represent the characteristics of a location. Such event topics are called *significant topics*. They will be the focus of this chapter to extract semantic annotations for locations. Identifying ETs for the first category, however, is not trivial. A naive method such as ranking ETs by counting their occurrences does not work, because the results will be dominated by very common, generic topics. That is, ETs of the second category will also be included. Thus, a suitable measure for the relevance of an ET is fundamental. Moreover, an ET might be significant only with respect to a given location at a particular time, e.g., on Saturdays. Therefore, temporal aspects need to be considered as well. Another challenge is that hierarchies exist not only for ETs but also for locations and time, and they need to be considered in deriving meaningful location annotations.

To tackle the above challenges, we propose a framework to extract location annotations from event data. For this, we introduce the concept of a Location-Time-pair Class (LTC) to describe a group of location-time pairs that have the same location and time concepts, e.g., ['*Stadium*','*Weekend*']. We define a measure to identify significant event topics with respect to an LTC, based on *Pointwise Mutual Information* [86]. A set of significant topics with respect to an LTC is called a Location-Time-pair Profile (LTP). LTPs are utilized further to derive semantic annotations for locations, where an annotation is a pair of an event topic and a time concept, e.g., ⟨'*Live-music*','*Weekend*'⟩. Figure 6.1 shows the components of our framework. *LTProfile-Miner* is an algorithm to extract LTPs from event datasets. Since extracting LTPs consumes most of the time in the overall process, running *LTProfile-Miner* whenever

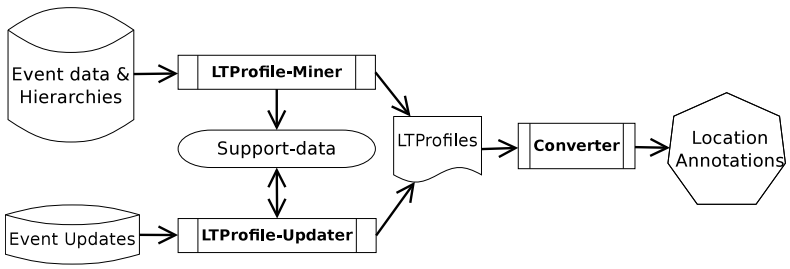


Figure 6.1: Conceptual framework for annotating event locations.

new events are added is very inefficient. Hence, to deal with periodic updates of the input dataset, we also present another algorithm, called *LTProfile-Updater*, to update the current set of location annotations. With the latter algorithm, we also provide a scalable, efficient and '*anytime*' approach to deal with large datasets that do not fit in main memory and take a long time to process. Here, '*anytime*' means the algorithm can be interrupted at any time with a valid solution and resumed later by using support-data stored on secondary storage.

Using external sources such as Wikipedia or Google Search, extracted annotations of the famous locations, e.g., stadiums, museums, or theatres, can be manually validated. Since there is no pre-existing ground-truth to validate all results obtained from a given dataset, we indirectly measure how good the extracted annotations are with location clustering.

In summary, the contributions of this chapter are as follows:

- We model semantic annotations for locations based on the concepts of events and event components as introduced in Chapter 3.
- We propose a measure based on *Pointwise Mutual Information* to identify significant ETs from event datasets.
- We develop an approach called *LTProfile-Miner* to derive location annotations from event datasets. An approach called *LTProfile-Updater* is used to efficiently deal with periodic updates of the datasets.
- We demonstrate the utility of extracted annotations in semantic location search and clustering by using real event data crawled from the Website *eventful.com*.

In the following section, we discuss related work. In Section 6.3, we introduce the basic concepts and notations. We describe our method to extract semantic annotations for locations in Section 6.4. After presenting experimental results in Section 6.5, we summarize the chapter in Section 6.6.

6.2 Related Work

Basically, the term '*annotation*' means to attach information (metadata) to existing data. An example is that Flickr users add tags to photos to describe what is shown or the meaning of photos. However, such human effort-based annotation systems as classified by a survey [6] are often noisy and incomplete.

Therefore, there have been many approaches to *automatically* annotate objects in different formats such as textual documents, photos or videos, e.g., [9, 23, 62]. However, extracting annotations from spatio-temporal data like event data raises many challenges, e.g., annotations might differ among regions as well as over time, as discussed in [29]. Thus, such approaches cannot be directly applied to extract location annotations from event datasets. Nevertheless, the idea of *word-context matrices* and a statistical measure successfully used in annotating textual documents, called *Pointwise Mutual Information* [86, 111], can be utilized to estimate correlations among locations, time, and event topics. This will be described in more detail in Section 6.3.2.

There are some approaches similar to our work in extracting annotations from spatio-temporal data. One direction of research relies on location information from external sources, such as the Google Maps API to annotate locations [5, 19, 21]. These approaches first extract points of interest (e.g., *stops* from trajectory data), and then annotate them with place categories (e.g., '*hotel*', '*touristic place*', or '*supermarket*') using external sources. Different from these approaches, we aim at extracting not only place categories but also relevant event topics with respect to a given location, important information that cannot be obtained from the above data sources.

Another direction of research aims at exploiting georeferenced social media to describe and annotate *geographic space*. Rattenbury and colleagues proposed several spatial clustering methods to identify Flickr tags corresponding to places and/or events [94, 95]. Such tags can then be used to annotate geographic space on the basis of discovered clusters. Similarly, the approach in [100] aims at extracting latent geographic place semantics from Flickr data. Geographic space can then be annotated using spatial distributions and coefficients of extracted features. Although the above approaches focus on extracting annotations from spatio-temporal data, they are only able to annotate geographic space in general and not specific locations. Furthermore, these approaches do not explicitly model locations, in particular, they do not consider location hierarchies.

To the best of our knowledge, little work has been done yet to annotate locations with semantic tags. One of the most related work is [128], where the authors propose a technique to annotate places with categorical tags such as '*restaurant*' or '*cinema*' by exploiting social network data. They build a support vector machine classifier for each predefined tag by learning from user check-in data with labeled places. Given an unlabeled place, the classifiers predict the missing tag to categorize the place. Similarly, the approach in [47] exploits check-in data to enrich places with semantic tags that are extracted from user interest-profiles on social networks. Since interest-profiles of users are often sparse and contain only a few keywords, they also extend interest profiles with related information based on Wikipedia links.

Since the approaches as mentioned above basically rely on characteristics of check-in data consisting of hidden user behaviors, they cannot be applied for event datasets for the following reasons. First, these approaches require a significant number of check-in records at a particular location and time to derive user behaviors. However, only few events occur at a particular location and time in an event dataset. The second reason is that a check-in record as described in their approaches is a triple $\langle user, time, location \rangle$ that does not contain semantic tags like an event description. Thus, in their approaches, a candidate set of tags for locations needs to be either predefined or obtained from an external source (user interest-profiles). Such a predefined set of tags is often small and only contains generic tags describing place categories, e.g., '*restaurant*' or '*cinema*'. Rather than focusing on generic tags describing place categories like the above approaches, we aim at extracting more informative tags that can be used to discriminate one location from another. Then, place categories can be derived later on using clustering. Finally, we also take concept hierarchies for time, locations and event topics into account, an important and useful piece of information not considered by the above approaches.

6.3 Basic Concepts and Notations

In the following section, we specialize the notations and definitions of events and event topics that utilize our event model for the purpose of extracting semantic annotations for locations. Section 6.3.2 introduces the concepts of a Location-Time-pair Instance and Class, and a measure based on normalized pointwise mutual information (npmi) to model Location-Time-Profiles. Finally, in Section 6.3.3, we detail how to compute npmi values from an event dataset.

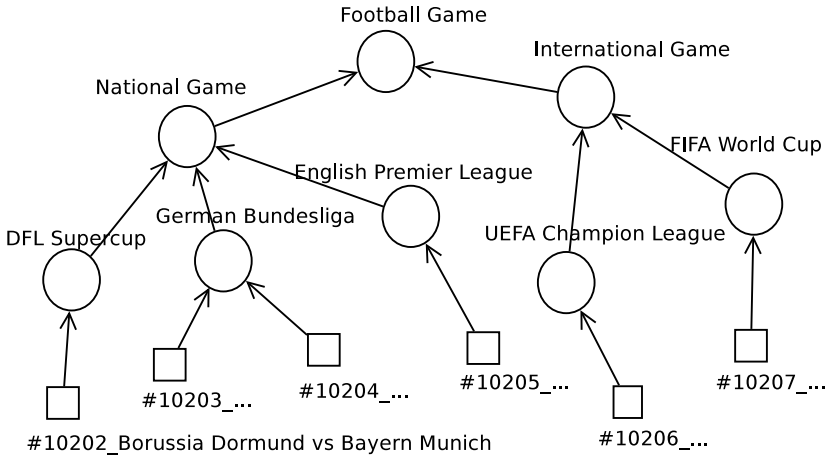


Figure 6.2: Example of a concept hierarchy (event topics) related to football games. Boxes represent event instances (event ID and names), circles represent higher level topics.

6.3.1 Events and Event Components

Recall that we model an event as a tuple $\langle e_{id}$, Context, Time, Location$\rangle$. The context, time, and location components of an event can be generalized to higher levels of abstraction and granularity, based on hierarchies. While Chapters 4 and 5 employ this event model for the discovery of patterns that describe spatial, temporal, and conceptual relationships among events, this chapter aims at the discovery of *significant correlations* among the *event components* at different levels of abstraction and granularity. Before describing our approach, we first specialize the context, location, and time frameworks introduced in Sections 3.2, 3.3.1, and 3.3.2, respectively, for the purpose of extracting semantic annotations for locations.

Briefly, the context component of an event represents a topic for that event. In data sources that manage information about public activities (e.g., festivals, sports, or concerts), an event context is typically provided as a textual description of the corresponding event, e.g., '*#10202_Borussia Dortmund vs Bayern Munich*' for a football game. A context can be generalized to higher levels of abstraction, based on a given concept hierarchy. For example, Figure 6.2 shows a simple hierarchy related to football, where the context '*#10202_Borussia Dortmund vs Bayern Munich*' can be generalized to '*DFL Supercup*', '*National Game*', and then '*Football Game*'. Such a hierarchy might be explicitly provided by the event data source, or it can be built using a learning approach, e.g., an approach described by Cimiano et al. [26].

We employ the operator $\Uparrow$ (for event contexts, defined in Section 3.2.3) to compute the set of all generalizations for a context in a given hierarchy. For example, '*DFL Su-*

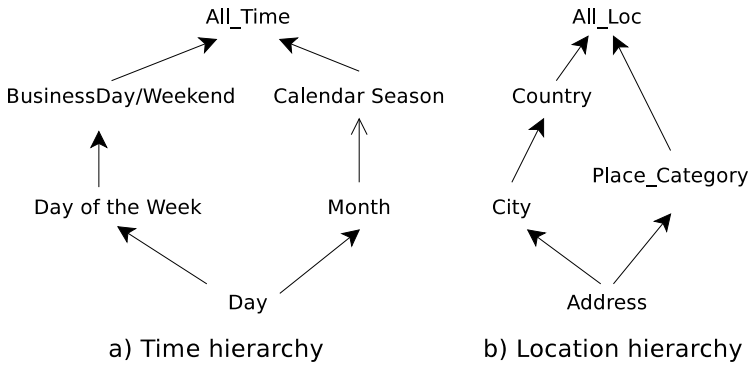


Figure 6.3: Hierarchies for locations and time.

percup'$^{\Uparrow}$ is the set {'*National Game*', '*Football Game*'} based on the hierarchy shown in Figure 6.2. In this chapter, event contexts and their generalizations are called *event topics* (ETs), which are key ingredients of semantic annotations for locations.

The time component of an event is typically specified as a time point (e.g., '*2013-06-27*') of a temporal granularity (e.g., *Day*). In this chapter, since we focus on events such as festivals, sports, or concerts, we assume that the time of an event is specified at (or can be mapped to) the granularity *Day*. By utilizing a predefined time hierarchy, an event time can be generalized to time concepts. For example, a time point '*2013-06-27*' can be generalized to '*Friday*' $\rightarrow$ '*BusinessDay*' $\rightarrow$ '*All_Time*', or '*June*' $\rightarrow$ '*Summer*' $\rightarrow$ '*All_Time*', based on the time hierarchy in Figure 6.3(a). We use the operator $\Uparrow$ (for time components of events, introduced Section 3.3.2) to compute the set of all generalizations of a time point. By applying this operator on the time point in the previous example, we obtain a set {'*Friday*', '*BusinessDay*', '*June*', '*Summer*', '*All_Time*'}.

Finally, the location component of an event describing where the event occurred is specified at a location granularity. Since an event like a concert or a football game takes place at a particular location, e.g., an arena or a stadium, we assume that locations of events are of granularity *Address*. They can be generalized to a coarser granularity like *City* or to a location concept (place category) like '*Stadium*', based on a predefined hierarchy. For example, based on the location hierarchy shown in Figure 6.3(b), a location '*Signal Iduna Park*' (a football stadium in Dortmund, Germany), is generalized as: '*Signal Iduna Park*' $\rightarrow$ '*Dortmund*' $\rightarrow$ '*Germany*' $\rightarrow$ '*All_Loc*', or '*Signal Iduna Park*' $\rightarrow$ '*Stadium*' $\rightarrow$ '*All_Loc*'. Note that '*Place_Category*' in Figure 6.3(b) might be replaced by a location taxonomy. For example, '*Signal Iduna Park*' can be generalized to '*Football venue in Germany*', '*Football venue*', and then '*Sports venue*', based on a taxonomy of locations, e.g., the Wikipedia categorization.

We again use the $\Uparrow$ operator to specify the generalizations of a location. For example, '*Signal Iduna Park*'$^{\Uparrow}$ is the set {'*Dortmund*', '*Germany*', '*Stadium*', '*All_Loc*'}, based on the hierarchy in Figure 6.3(b).

Given an event topic f[1](e.g., '*Football Game*'), a time concept T (e.g., '*Weekend*'), and a location L (e.g., '*Stadium*'), one might find some events whose respective components are related to f, T, and L from a given event dataset. The more such events are found, the more significant the association of f, T, and L is. In reality, some associations are more significant than others. For example, it is more likely to find events related to ice skating in Winter than in Summer, or it is more likely to find a rock festival in some cities (such as Nürburg or Munich) than in other cities. To model such associations, we introduce the concepts of a *Location-Time-pair Instance* and a *Location-Time-pair Class* in the following section.

6.3.2 Location-Time-pair Instances and Classes

Using the above notations of events and event topics, we now introduce the concepts of a *Location-Time-pair Instance* and *Class* to model *Location-Time-Profiles*.

Let $\mathcal{D}$ be a dataset of events as $\langle e_{id}$, Context, Time, Location$\rangle$ tuples, where the time and location components of each event are of granularity *Day* and *Address*, respectively. To formulate the probability to find an event topic at a given location and time later on, we define a *Location-Time-pair Instance* (LTI) as a pair $[l, t]$, where l and t are the location and time of some event in $\mathcal{D}$. The set of LTIs with respect to a given dataset $\mathcal{D}$ of events is defined as below.

Definition 6.1 (LTI Set) *Given a dataset $\mathcal{D}$ of events, the **LTI set** of $\mathcal{D}$ is defined as:*

$$\mathcal{I}(\mathcal{D}) := \{[l, t] \mid \exists e \in \mathcal{D}, e.Location = l \wedge e.Time = t\}. \tag{6.1}$$

Given an LTI $[l, t] \in \mathcal{I}(\mathcal{D})$, $\mathcal{D}[l, t]$ denotes a subset of $\mathcal{D}$ where the location and time of each event in $\mathcal{D}[l, t]$ are l and t, respectively, i.e.,

$$\mathcal{D}[l, t] := \{e \in \mathcal{D} | e.Location = l \wedge e.Time = t\}. \tag{6.2}$$

As mentioned before, a location l can be generalized to a concept L based on a given location hierarchy, e.g., '*Signal Iduna Park*' $\rightarrow$ '*Stadium*' $\rightarrow$ '*Sports-venue*'. Similarly, a time point t can be generalized to a time concept T, based on a time hierarchy, e.g., '*2013-07-28*' $\rightarrow$ '*Sunday*' $\rightarrow$ '*Weekend*'. The pair $[L, T]$ is called a

[1]In this chapter, we often use 'e' to denote an event and 'f' to denote an event topic.

Location-Time-pair Class (LTC) and the LTI $[l, t]$ is called an *instance* of that LTC. Continuing the examples above, one can infer that ['*Signal Iduna Park*', '*2013-07-28*'] is an instance of ['*Stadium*', '*Weekend*']. A formal definition of LTCs is as follows.

Definition 6.2 (LTC Set) *Given a dataset $\mathcal{D}$ of events, a time hierarchy, and a location hierarchy, the* ***LTC set*** *of $\mathcal{D}$ is defined as:*

$$\mathcal{C}(\mathcal{D}) := \{[L, T] \mid \exists [l, t] \in \mathcal{I}(\mathcal{D}), L \in l^{\Uparrow} \wedge T \in t^{\Uparrow}\}. \tag{6.3}$$

Given an LTC $\Omega = [L, T] \in \mathcal{C}(\mathcal{D})$, an LTI $\omega = [l, t] \in \mathcal{I}(\mathcal{D})$ is an ***instance*** *of the LTC Ω iff $L \in l^{\Uparrow}$ and $T \in t^{\Uparrow}$.*

Given an LTC, it is straightforward to retrieve the set of its instances (LTIs). For each LTI, event topics can then be generated from events in that LTI. Therefore, it is reasonable to determine the correlation between a given LTC and an ET by analyzing the occurrences of that ET in the LTC. Clearly, ETs that are strongly related to an LTC are important to represent the characteristics of that LTC. For example, topics such as '*football*', '*sport*', or '*celebration*' are expected for the LTC ['*Stadium', 'Weekend*'], whereas '*drink*', '*live music*', or '*shows*' are expected for the LTC ['*Bar/Club*','*Weekend*'].

To formulate the correlations as mentioned above, we borrow an idea from Computational Linguistics to compute the correlation between a word and a context [25, 111]. Here, we consider an LTC a word and an ET a context. A word-context matrix (here called LTC-ET matrix) has the following properties:

(1) Each row corresponds to an LTC.

(2) Each column corresponds to an ET.

(3) The value of the element at a row Ω and a column f is the probability to find an LTI of the LTC Ω containing an instance (event) of the ET f. In other words, it is the joint probability P(f,Ω).

A measure to compute the correlation between a word and a context, called *Pointwise Mutual Information (pmi)*, is proven to work well to compute semantic similarities among words [86]. Also considering semantic similarities among locations later on, we employ this measure to compute the correlation between an LTC and an ET. The *pmi* value for an ET f with respect to an LTC Ω is computed from the two following probabilities:

(1) the probability to find f at any instance (LTI) of Ω, i.e., the conditional probability $\mathrm{P}(f|\Omega)$, and

(2) the probability to find f at any LTI in the dataset, i.e., $\mathrm{P}(f)$

The measure is defined as below.

Definition 6.3 *(**Pointwise Mutual Information**) Given an ET f and an LTC Ω, the **pointwise mutual information (pmi)** of f and Ω is defined as:*

$$pmi(f;\Omega) := \log\left(\frac{P(f|\Omega)}{P(f)}\right) = \log\left(\frac{P(f,\Omega)}{P(f)P(\Omega)}\right). \tag{6.4}$$

In the above definition, a *pmi* value of an ET f with respect to an LTC Ω represents the logarithmic difference between the two probabilities $\mathrm{P}(f|\Omega)$ and $\mathrm{P}(f)$. Thus, the *pmi* can be zero, positive or negative. If it is zero, i.e., $\mathrm{P}(f|\Omega) = \mathrm{P}(f)$, f and Ω are independent. If the value is positive, i.e., $\mathrm{P}(f|\Omega) > \mathrm{P}(f)$, the events related to f occur more likely at Ω than at other LTCs. If the value is negative, i.e., $\mathrm{P}(f|\Omega) < \mathrm{P}(f)$, the events related to f more rarely occur at Ω than at other LTCs.

The *pmi* measure can be normalized to a value between [-1,+1], where -1 means negatively correlated, 0 for independence, and +1 for perfectly correlated [13].

Definition 6.4 *(**Normalized Pmi**) Given an event topic f and an LTC Ω, the **normalized pointwise mutual information (npmi)** of f and Ω is defined as:*

$$npmi(f;\Omega) := \frac{pmi(f;\Omega)}{-\log(P(f,\Omega))} \qquad \in [-1,1]. \tag{6.5}$$

As shown in Formulas (6.4) and (6.5), the *npmi* measure represents the difference between the probabilities $\mathrm{P}(f|\Omega)$ and $\mathrm{P}(f)$. Therefore, the *npmi* measure typically gives an ET a high score with respect to a given LTC if the ET frequently occurs at that LTC but rarely at other LTCs. For example, in our experiments with sports events crawled from the Website *eventful.com*, the topics '*borussia*' or '*bundesliga*' get higher *npmi* scores than the topics '*football*' or '*soccer*' with respect to an LTC ['*Signal Iduna Park*', '*Weekend*'][2]. One can see that the first three topics are better to identify that LTC, and thus, they have priority over the last two topics to annotate the location '*Signal Iduna Park*'. In addition, if a rare ET is found only at a given LTC, the *npmi* measure also gives the ET a high score with respect to that LTC,

[2]Signal Iduna Park is the home stadium of the Borussia Dortmund football team playing in the German Bundesliga.

even if the ET contains very few instances. Such ETs are common in real event datasets, for instance, annual festivals at a particular location (e.g., '*Oktoberfest*' in Munich - Germany, or '*Puccini Festival*' in Torre del Lago - Italy). This is different from using a frequency-based measure such as the *tf-idf* (term frequency - inverse document frequency). Although the measure *tf-idf* increases the score of a rare ET with the *idf* factor, it still gives low scores for very rare ETs because of the *tf* factor.

Another advantage of the *npmi* measure is as follows. Since a frequency-based measure like the *tf-idf* always gives a non-negative value, it is not trivial for the user to pick a good threshold in order to filter out irrelevant ETs. On the other hand, a non-positive *npmi* value indicates an insignificant correlation between an ET and an LTC, thus, one can use any positive threshold δ to filter out irrelevant ETs (whose *npmi* values are zero or negative). With a positive threshold δ, one can select only ETs that have significant correlations to a given LTC. A set of such event topics is called a *Location-Time-Profile*.

Definition 6.5 *(**Location-Time-Profile**) Let $\mathcal{D}$ be a dataset of events and Ω be an LTC in $\mathcal{C}(\mathcal{D})$. The **profile** of Ω with respect to a given threshold $\delta > 0$ is a set of ETs, defined as:*

$$Profile(\Omega) := \{f \in e.Context^{\Uparrow} |\ e \in \mathcal{D} \wedge npmi(f; \Omega) \geq \delta\}. \tag{6.6}$$

For a particular purpose such as location clustering where feature selection can be viewed as a form of weighting, both the *npmi* and *tf-idf* can be used. However, as shown in our experiments later, the *npmi* measure performs better than *tf-idf* when considering semantic similarity between locations.

In the next section, we present a method to compute the *npmi* from the observed events in the input dataset $\mathcal{D}$.

6.3.3 NPMI Estimation

To estimate the probabilities in Formulas 6.4 and 6.5, we define *support* of an event topic.

Definition 6.6 *(**Support**) Let $\mathcal{D}$ be a dataset of events and f be an ET. An LTI $[l, t]$ **supports** f iff there exists an event $e \in \mathcal{D}[l, t]$ such that e is an instance of f.*

For example, because the event $e = \langle$'*#10202*','*Borussia Dortmund vs Bayer Munich*', '*2013-06-27*', '*Signal Iduna Park*'$\rangle$ is an instance of an ET $f =$'*Football Game*', one can infer that ['*Signal Iduna Park*', '*2013-06-27*'] supports f.

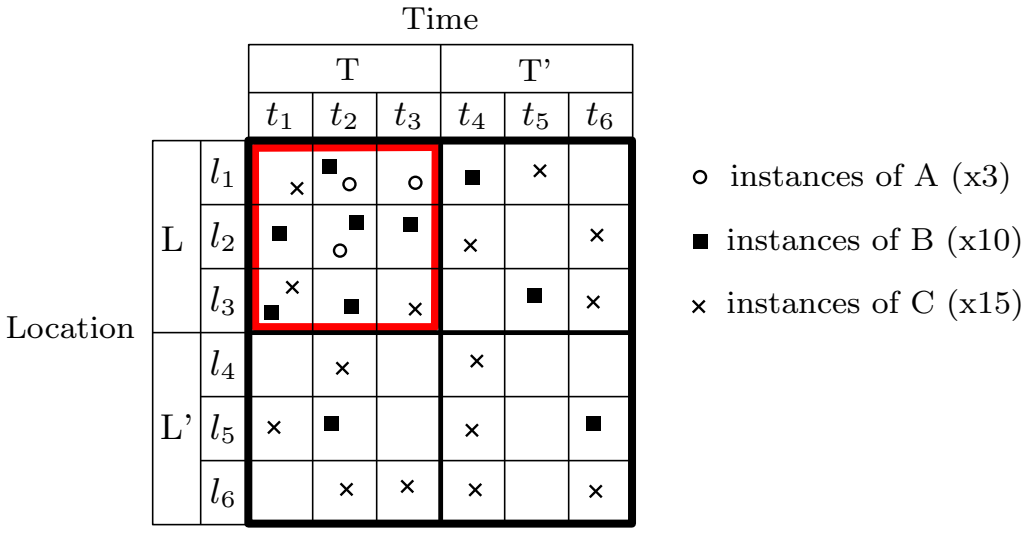


Figure 6.4: A toy dataset to illustrate the *npmi* measure. There are 6 locations (l_1, l_2, ..., l_6) and 6 time points (t_1, t_2, ..., t_6) in the dataset. The locations l_1, l_2, l_3 are generalized to a location concept L; the other locations are generalized to a location concept L'. The time points t_1, t_2, t_3 are generalized to a time concept T; the other time points are generalized to a time concept T'. The LTC of interest is marked as a box. A, B, C are event topics, where all instances of A ($\frac{3}{3}$), 60% instances of B ($\frac{6}{10}$), and 20% instances of C ($\frac{3}{15}$) occur inside the box.

Given an event dataset $\mathcal{D}$, an ET f, and an LTC Ω, we now estimate the probabilities P(f,Ω), P(f), and P(Ω) in Formulas 6.4 and 6.5.

Let N be the size of the LTI set of $\mathcal{D}$ (i.e., $N = |\mathcal{I}(\mathcal{D})|$), N_f be the number of LTIs in $\mathcal{I}(\mathcal{D})$ that support f, N_Ω be the number of LTIs in $\mathcal{I}(\mathcal{D})$ that are instances of Ω, and $N_{f,\Omega}$ be the number of instances of Ω in $\mathcal{I}(\mathcal{D})$ that support f. The probabilities P(f,Ω), P(f), and P(Ω) are estimated as:

$$P(f,\Omega) = \frac{N_{f,\Omega}}{N}, \quad P(f) = \frac{N_f}{N} \quad and \quad P(\Omega) = \frac{N_\Omega}{N}.$$

Thus,

$$pmi(f;\Omega) = \log\left(\frac{P(f,\Omega)}{P(f)P(\Omega)}\right) = \log\left(\frac{\frac{N_{f,\Omega}}{N}}{\frac{N_f}{N} * \frac{N_\Omega}{N}}\right) = \log\left(\frac{N_{f,\Omega} * N}{N_f * N_\Omega}\right)$$

and

$$npmi(f;\Omega) = \frac{pmi(f;\Omega)}{-\log(P(f,\Omega))} = \frac{\log\left(\frac{N_{f,\Omega}*N}{N_f*N_\Omega}\right)}{-\log\left(\frac{N_{f,\Omega}}{N}\right)} = \frac{\log\left(\frac{N_{f,\Omega}*N}{N_f*N_\Omega}\right)}{\log\left(\frac{N}{N_{f,\Omega}}\right)}. \tag{6.7}$$

We illustrate the *npmi* measure by using a toy dataset shown in Figure 6.4. One can see that in comparison to instances of the event topic B, instances of event topics

A and C are more likely to occur at a location and time related to the LTC $\Omega = [L, T]$ (marked as a box). We now show how to verify this fact with the *npmi* measure. For this, we compute the *npmi* values for A, B, and C with respect to the LTC Ω, as shown below.

$$npmi(A;\Omega) = \frac{\log\left(\frac{N_{A,\Omega}*N}{N_A*N_\Omega}\right)}{\log\left(\frac{N}{N_{A,\Omega}}\right)} = \frac{\log\left(\frac{3*36}{3*9}\right)}{\log\left(\frac{36}{3}\right)} = 0.56$$

$$npmi(B;\Omega) = \frac{\log\left(\frac{N_{B,\Omega}*N}{N_B*N_\Omega}\right)}{\log\left(\frac{N}{N_{B,\Omega}}\right)} = \frac{\log\left(\frac{6*36}{10*9}\right)}{\log\left(\frac{36}{6}\right)} = 0.49$$

$$npmi(C;\Omega) = \frac{\log\left(\frac{N_{C,\Omega}*N}{N_C*N_\Omega}\right)}{\log\left(\frac{N}{N_{C,\Omega}}\right)} = \frac{\log\left(\frac{3*36}{15*9}\right)}{\log\left(\frac{36}{3}\right)} = -0.09$$

Clearly, the values of $npmi(A;\Omega)$ and $npmi(B;\Omega)$ are positive whereas the value of $npmi(C;\Omega)$ is negative. Moreover, since the value of $npmi(A;\Omega)$ is larger than the value of $npmi(B;\Omega)$, one can infer that the correlation between the event topic A and the LTC Ω is stronger than the correlation between the event topic B and the LTC Ω. This is reasonable because instances of A can be found only at a location and time related to the LTC Ω while instances of B are sometimes found at a location and time not related to the LTC Ω. The event topic A in this example is an illustration of rare events that can be found only at a particular location and time.

In conclusion, the *npmi* is useful to determine the correlation between an ET and an LTC, and a positive *npmi* threshold can be used to filter out insignificant ETs with respect to a given LTC.

6.4 LT-Profiles and Applications

We now introduce our approach to generate LT-Profiles from an event dataset, and a scalable and efficient method to deal with periodic updates of the input data. We then show how to convert such profiles into location annotations. Finally, we describe how to exploit such information in semantic location search and clustering.

6.4.1 Generating Location-Time-Profiles

Given a dataset $\mathcal{D}$ of events, a set $\mathcal{H}$ of hierarchies for generating ETs from events and for generating location and time concepts, and a *npmi* threshold δ, this section describes a procedure to find all profiles as defined in Definition 6.5.

The process of generating profiles for LTCs consists of two steps: (1) generate LTCs and their ET candidates, and (2) for each LTC, filter out ETs in the candidate set whose *npmi* value does not satisfy the threshold.

Algorithm 6.1: LTProfile-Miner

Input:
(a) D: event dataset
(b) $\mathcal{H}$: set of hierarchies (with support of $\Uparrow$ operation)
(c) δ: npmi threshold
Output: set of all LTPs

```
   /* Step 1:  Generate LTCs and ET candidates                              */
1  LTI_Set = {[e.Location, e.Time] | e ∈ D} ;
2  LTC_Set = {};
3  foreach LTI ω = [l, t] ∈ LTI_Set do
4      foreach LTC Ω = [L, T] ∈ l^⇑ × t^⇑ do
5          if Ω ∉ LTC_Set then
6              LTC_Set = LTC_Set ∪ {Ω};
               //  initialize the set of ET candidates for Ω
7              Candidates[Ω] = {};
8          Candidates[Ω] = Candidates[Ω] ∪ {f ∈ e.Context^⇑ | e ∈ E[l, t]};
   /* Step 2:  Filter out insignificant ETs                                */
9  compute the support set for each ET f ∈ Candidates (Support_ET[f]);
10 compute the support set for each LTC Ω ∈ LTC_Set (Support_LTC[Ω]);
11 initialize the list Profile[ ];
12 N =| LTI_Set |;
13 foreach Ω ∈ LTC_Set do
14     Profile[Ω] = {};
15     foreach f ∈ Candidates[Ω] do
16         N_f = | Support_ET[f] |;
17         N_Ω = | Support_LTC[Ω] |;
18         N_{f,Ω} = | Support_ET[f] ⋂ Support_LTC[Ω] |;
19         npmi=log((N_{f,Ω}*N)/(N_f*N_Ω))/log(N/N_{f,Ω}) ;
20         if npmi ≥ δ then
21             Profile[Ω] = Profile[Ω] ∪ {f};
22 return Profile;
```

Lines 1-8 in Algorithm 6.1 show the pseudocode of the first step. In Line 1, a set of LTIs (*LTI_Set*) is computed by scanning all events in the dataset $\mathcal{D}$. The set of LTCs (*LTC_Set*) are then generated from these LTIs, as shown in Lines 3-6. For each

Table 6.1: Descriptions of the variables used in Algorithm 6.1

Variable	Type	Description
LTI_Set	set of LTIs	set of all LTIs extracted from the input dataset D ($\mathcal{I}(\mathcal{D})$)
LTC_Set	set of LTCs	set of all LTCs derived from the LTI_Set
$Candidates[\Omega]$	set of ETs	set of topic candidates for each LTC $\Omega \in LTC\_Set$ ($\mathcal{C}(\mathcal{D})$)
$Support\_ET[f]$	set of LTIs	set of LTIs that support an ET $f \in Candidates$
$Support\_LTC[\Omega]$	set of LTIs	set of LTIs that are instances of an LTC $\Omega \in LTC\_Set$
$Profile[\Omega]$	set of ETs	set of significant ETs for each LTC $\Omega \in LTC\_Set$
N	numeric	number of all LTIs in the input dataset
N_f	numeric	number of LTIs that support an ET $f \in Candidates$
N_Ω	numeric	number of instances of an LTC $\Omega \in LTC\_Set$
$N_{f,\Omega}$	numeric	number of instances of Ω that support f

LTC Ω derived from an LTI ω, a set of ET candidates ($Candidates[\Omega]$) is generated from the set of events in the LTI ω (Line 8).

To filter out insignificant ETs with respect to an LTC, the *npmi* of an LTC and an ET candidate needs to be computed. A naive method is to consider separately each LTC-ET pair. However, such a method is inefficient since the LTI_Set will be scanned multiple times for counting LTIs to compute the *npmi* by employing Equation (6.7). Thus, to compute *npmi* values efficiently, we use two data structures, called $Support\_ET$ and $Support\_LTC$, where each one is a hash table mapping keys to LTI sets. The first hash table ($Support\_ET$) maps ETs to LTI sets. Given an ET f, the set of LTIs that support f is retrieved by using the hash table $Support\_ET$. This set is denoted $Support\_ET[f]$. The second hash table ($Support\_LTC$) maps LTCs to LTI sets. Similarly, given an LTC Ω, $Support\_LTC[\Omega]$ is a set of LTIs that are instances of the LTC Ω. As shown in the pseudocode, the hash tables $Support\_ET$ and $Support\_LTC$ are computed in Lines 9 and 10. Utilizing these data structures allows the *npmi* for each pair of an LTC and an ET candidate to be computed with several set operations, as shown in Lines 16-19. Finally, the profile of each LTC Ω in the LTC_Set is computed by selecting all ETs in the candidate set whose *npmi* values satisfy the threshold δ (Lines 20-21).

The (runtime) complexity of Algorithm 6.1 is closely related to the characteristic of the dataset $\mathcal{D}$. In particular, it depends on the following elements:

1. number of events (n_e) in $\mathcal{D}$,
2. number of LTIs (n_i) that are derived from the events,
3. average number of events (n_{ei}) in an LTI,
4. total number of LTCs (n_c) that are derived from the LTIs,

5. average number of LTCs (n_{ci}) that are derived from an LTI,

6. number of ETs (n_f) that are generalized from the events,

7. average number of ETs (n_{fc}) generalized from the events of an LTC.

By using these variables, we analyze the complexity of Algorithm 6.1 as detailed below.

The complexity of Algorithm 6.1 consists of five components: (1) generate *LTI_Set* in Line 1, (2) generate the *LTC_Set* and ET candidates in Lines 3-8, (3) compute *Support_ET* in Line 9, (4) compute *Support_LTC* in Line 10, and (5) filter out insignificant ETs in Lines 13-21. They are individually analyzed in detail as follows.

Clearly, the complexities of the components (1) and (2) are $O(n_e)$ and $O(n_i n_{ci} n_{fc})$, respectively. The complexity to build *Support_ET* (the third component) is $O(n_f n_i n_{ei})$, since each element (corresponding to an ET) is computed by scanning through all the LTIs and considering all events inside each LTI. The complexity to build *Support_LTC* (the fourth component) is $O(n_c n_i)$, with an assumption that checking generalization relationships between an LTI and an LTC can be done with one instruction.

With the two data structures in Lines 9 and 10, the time to compute each *npmi* depends on only a few set operations, and it can be considered small if sets are implemented as hash sets. Therefore, the complexity of the fifth component is $O(n_c n_{fc})$. Thus, the overall complexity of the algorithm is $O(n_e + n_i n_{ci} n_{fc} + n_f n_i n_{ei} + n_c n_i + n_c n_{fc})$. In real datasets, the number of events n_e, the number of LTIs n_i and the number of LTCs n_c are much larger than the other variables. Thus, the complexity can be simplified to $O(n_e + n_i n_c)$.

6.4.2 Updating Location-Time-Profiles

In the previous section, we presented an algorithm to mine *LTProfiles* from a given dataset of events. Such a dataset consists of events in a certain time-interval (e.g., [2011,2012]), thus, the extracted profiles are only valid in this interval. In reality, datasets are incrementally updated. For example, events in 2013 are added to a dataset of events in [2011,2012]. Running again *LTProfile-Miner* for the merged dataset is a possible solution, which, however, is neither efficient nor scalable. To adapt to periodic updates of event data, we propose another algorithm, called *LTProfile-Updater*.

Algorithm 6.2: LTProfile-Updater

Input:
(a) D^*: dataset of new events to update (event delta)
(b) $\mathcal{H}^*$: set of hierarchies for the events in D^*
(c) δ: npmi threshold

Output: set of all LTPs

Load the following variables from a secondary storage: $\{LTC\_Set, Candidates, Support\_ET\_Size, Support\_LTC\_Size, Support\_LTC\_ET\_Size, N\}$;
`/* Step 1: Update the` LTC_Set `and` $Candidates$ `*/`
$LTI\_Set^* = \{[e.Location, e.Time] \mid e \in \mathcal{D}^*\}$;
foreach $LTI\ \omega = [l, t] \in LTI\_Set^*$ **do**
  foreach $LTC\ \Omega = [L, T] \in l^{\Uparrow} \times t^{\Uparrow}$ **do**
    if $\Omega \notin LTC\_Set$ **then**
      $LTC\_Set = LTC\_Set \cup \{\Omega\}$;
      `//  initialize the set of ET candidates for` Ω
      $Candidates[\Omega] = \{\}$;
    $Candidates[\Omega] = Candidates[\Omega] \cup \{f \in e.Context^{\Uparrow} \mid e \in E[l, t]\}$;
`/* Step 2: Filter out insignificant ETs */`
compute $Support\_ET^*[f]$ for each ET f, and $Support\_LTC^*[\Omega]$ for each LTC Ω from $\mathcal{D}^*$ and $\mathcal{H}^*$;
initialize the list $Profile[\ ]$;
$N = N + |LTI\_Set^*|$;
foreach $\Omega \in LTC\_Set$ **do**
  $Profile[\Omega] = \{\}$;
  foreach $f \in Candidates[\Omega]$ **do**
    $N_f = Support\_ET\_Size[f] + \mid Support\_ET^*[f] \mid$;
    $N_\Omega = Support\_LTC\_Size[\Omega] + \mid Support\_LTC^*[\Omega] \mid$;
    $N_{f,\Omega} = Support\_LTC\_ET\_Size[f, \Omega] + \mid Support\_ET^*[f] \bigcap Support\_LTC^*[\Omega] \mid$;
    $npmi = log(\frac{N_{f,\Omega} N}{N_f N_\Omega}) / log(\frac{N}{N_{f,\Omega}})$;
    if $npmi \geq \delta$ **then**
      $Profile[\Omega] = Profile[\Omega] \cup \{f\}$;
    $Support\_ET\_Size[f] = N_f$;
    $Support\_ET\_Size[\Omega] = N_\Omega$;
    $Support\_LTC\_ET\_Size[f, \Omega] = N_{f,\Omega}$;
Store the following variables to a secondary storage: $\{LTC\_Set, Candidates, Support\_ET\_Size, Support\_LTC\_Size, Support\_LTC\_ET\_Size, N\}$;
return $Profile$;

Assume that after executing *LTProfile-Miner* (Algorithm 6.1), the following intermediate values are stored on secondary storage: LTC_Set, $Candidates$, N_f (as an element of a list $Support\_ET\_Size[f]$), N_Ω (as an element of a list $Support\_LTC\_Size[\Omega]$), $N_{f,\Omega}$ (as an element of a matrix $Support\_LTC\_ET\_Size$[f,Ω]), and N. One can find the descriptions of these values in Table 6.1. Such data, called *support-data*, contain sufficient information to extract profiles without considering the original (previous) dataset $\mathcal{D}$.

As shown in Line 1 of Algorithm 6.2, *LTProfile-Updater* first loads the *support-data* and then combines it with the update ($\mathcal{D}^*$, $\mathcal{H}^*$) to update the current location profiles. Note that in Algorithm 6.2, we use '*' to denote values computed from the update. Basically, *LTProfile-Updater* is similar to *LTProfile-Miner*, but it utilizes the support-data so that only the update $\mathcal{D}^*$ is scanned (Line 2-8).

It is reasonable to assume that each event in $\mathcal{D}^*$ occurred after all events in $\mathcal{D}$, i.e., events in $\mathcal{D}^*$ are newer than events in $\mathcal{D}$. Therefore, there is no overlap between the LTI sets of the two datasets. Thus, the values of N, N_f, N_Ω and $N_{f,\Omega}$ can be computed as shown in Line 11-17. For future updates, such variables are also updated and stored on secondary storage (Line 21-24).

In Algorithm 6.2 (Lines 18-20), one can see that the *npmi* of each LTC-ET pair is recomputed and compared to the threshold δ. Therefore, with new events from the update ($\mathcal{D}^*$, $\mathcal{H}^*$), *LTProfile-Updater* might add new ETs to a profile and remove from it all ETs that become irrelevant. In other words, the profiles that are previously computed might change after updating. Consequently, the annotations derived from these profiles are also updated. Although the algorithm *LTProfile-Updater* is designed for the purpose of making annotations of locations up-to-date, it is trivial to extend this algorithm to also maintain the history of annotations for each location.

By employing *LTProfile-Updater*, an *anytime* approach to deal with very large datasets works as follows. First, the events in a (large) dataset $\mathcal{D}$ are sorted by the time attribute and partitioned in increasing order into sub-datasets $\mathcal{D}_0$, $\mathcal{D}_1$, $\mathcal{D}_2$,... such that each $\mathcal{D}_i$ fits into main memory. *LTProfile-Miner* is then called to compute the support-data from $\mathcal{D}_0$. Finally, *LTProfile-Updater* is iteratively called for each $\mathcal{D}_i$ ($i \geq 1$). If the mining process is interrupted after processing $\mathcal{D}_i$, the results are valid until the latest time in $\mathcal{D}_i$.

Since the complexity of *LTProfile-Updater* and *LTProfile-Miner* can be analyzed in an analogous way, we omit the complexity analysis for *LTProfile-Updater* here.

6.4.3 Location Annotations

In the previous sections, we presented efficient methods to extract and update Location-Time-Profiles from event datasets. In this section, we describe how to utilize such profiles in annotating locations.

Location-Time-Profiles, each consisting of significant ETs at an LTC, can be exploited to annotate locations. Here, we define a location annotation as a set, where each element is a pair of an event topic and a time concept. For example, annotation elements for a bar/club might be $\langle jazz, Tuesday \rangle$, $\langle live\_music, Weekend \rangle$ or $\langle dancing, All\_Time \rangle$. We now give a formal definition of location annotations.

Definition 6.7 *(**Location Annotation**) Let $\mathcal{D}$ be an event dataset. The **annotation** of a location (or location concept) L is a set defined as:*

$$Annotation(L) := \{\langle f, T \rangle \mid \Omega = [L, T] \in \mathcal{C}(\mathcal{D}),\ f \in Profile(\Omega)\}. \tag{6.8}$$

Once the set of profiles is generated, generating annotation for a location L is straightforward. First, a set of all profiles related to L (i.e., their LTCs are of the form [L,*]) is selected. The annotation of L is a set consisting of elements of the form of $\langle f, T \rangle$, where T is a time concept and f is an ET in the profile of the LTC $[L, T]$.

From a given dataset of events, all location annotations are generated in two steps. First, profiles are generated with Algorithm 6.1 or updated with Algorithm 6.2. Next, for each location appearing in some LTC, its annotation is generated by the process as described above. In the next section, we will describe how to exploit such annotations in semantic location search and clustering.

6.4.4 Similarity Measure for Locations

Typically, a location-based service manages information about a large number of locations. To efficiently categorize and organize these locations, one needs a measure to determine how similar two locations are. For this, we define a similarity measure for locations based on events. Basically, the more common event topics two locations have, the more similar they are. We detail this idea in the following.

Given two locations L_1 and L_2, and their annotations AL_1 and AL_2, respectively, the similarity between the two locations is defined by using the Jaccard Index as:

$$sim(L_1, L_2) := \frac{|AL_1 \cap AL_2|}{|AL_1 \cup AL_2|} \quad \in [0, 1]. \tag{6.9}$$

This measure allows one to find locations that are similar to a given location or just to rank the results, for example in the query "*Find all cities in the US like Munich (in Germany) in terms of festivals*". For such a query, the system typically provides a graphical user interface that enables the user to specify a location of interest (e.g., Munich), an event topic of interest (e.g., festival), and spatial/temporal constraints (e.g., in the US, or from 2010 to 2012).

6.4.5 Location Clustering based on Annotations

Cluster analysis is a useful task to organize a large number of objects [43]. In this section, we utilize location annotations to cluster locations. Our goal is to cluster locations on the basis their semantic similarity. For example, stadiums are expected to be the same cluster. For this, we define a (semantic) distance function between two locations L_1 and L_2 as

$$dist(L_1, L_2) := 1 - sim(L_1, L_2), \tag{6.10}$$

where $sim(L_1, L_2)$ is the similarity between the locations L_1 and L_2, computed by employing Equation 6.9. With this distance function, locations can then be clustered with one of the various clustering algorithms such as hierarchical clustering.

In Section 6.3.1, we assumed that locations can be generalized to location concepts based on a given simple taxonomy (e.g., *'Signal Iduna Park'* $\rightarrow$ *'Stadium'*). To build a more complex taxonomy for locations, a hierarchical clustering method is used, and then labels of clusters are automatically generated from the set of annotations. For example, a label might be a combination of the top-k most frequent ETs of annotations in the corresponding cluster. These labels can be refined further by the user, for example, a cluster of locations related to *'football'*, *'soccer'* and *'bundesliga'* might be labeled as *'German football stadium'*. Following that, taxonomies of locations are generated. Such taxonomies can be encoded in RDF and linked to other data sources, e.g., point-of-interest (POI) data, to enrich them.

6.5 Experimental Evaluation

We demonstrate the utility and efficiency of our approach using datasets crawled from the Website *eventful.com* for different topics from 2011 to 2012. Our framework is implemented in Java and runs with 24GB heap size. All experiments were run on an

Table 6.2: Properties of datasets used in experiments.

Dataset	Topic	Area	Number of Events			Number of Locations
			2011	2012	Total	
DE-Sports	sports	Germany	1,335	1,673	3,008	960
DE-Festival	festival	Germany	1,278	1,654	2,932	1,515
EU-Festival	festival	Europe	13,592	20,561	34,143	18,018
DE-Music	music	Germany	24,756	32,398	57,154	12,591
DE-All	all topics	Germany	72,672	85,995	158,667	20,141

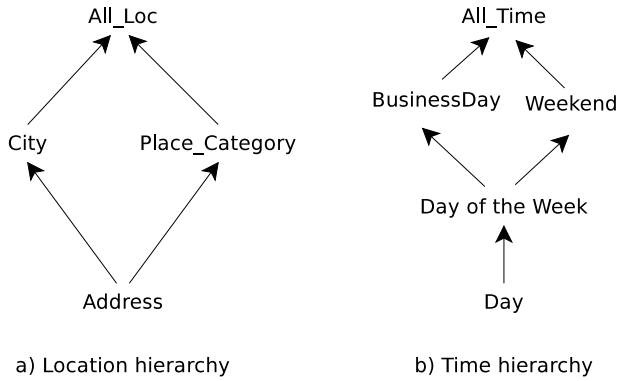


Figure 6.5: Hierarchies for locations and time used in the experiments.

Intel Xeon 2.27GHz with 48GB RAM, running Ubuntu 64bit. In the following, we first describe the setup of our experiments and then present the experimental results.

6.5.1 Datasets and Experimental Setup

To determine how good an annotation extracted for a particular location is, one needs background knowledge about that location. Therefore, we consider events in Germany and only festivals (as one event topic) in Europe in our experiments to easily validate the results later on. These events were crawled by using the eventful.com API. As raw data, each event consists of an event identifier, title, time, location, and a list of tags. Tags can be considered keywords attached to events to categorize events. Based on tags, one can select events for a particular topic, e.g., '*sport*', '*festival*', or '*music*'.

As mentioned in the complexity analysis, the runtime complexity of Algorithm 6.1 depends on not only the number of events but also the number of locations (more precisely, LTIs). Hence, for evaluation purposes, we consider different datasets of various topics and sizes in terms of the number of events and the number of locations. Table 6.2 shows the datasets used in our experiments, where the first two datasets (DE-Festival and DE-Sports) are smaller than the last three. All events took place in Germany or Europe in the years 2011 and 2012.

Similar to the experiments presented in the previous chapter, the raw data of events are transformed into the form $\langle e_{id}, Context, Time, Location \rangle$, where e_{id} is the event identifier and the last three components are the following attributes: the event identifier[3], start-time, and venue identifier, respectively. Following that, the context of an event can be generalized to higher levels of abstraction by utilizing the mapping from the set of event identifiers to the set of tags. We also employ the hierarchy for tags described in the previous chapter (see Section 5.6.1 - page 101).

For locations, we use the hierarchy as shown in Figure 6.5(a), and we aim at extracting annotations for locations at the following levels of abstraction: *Address*, *City*, and *Place_Category*. For time, we use the hierarchy as shown in Figure 6.5(b), where the time component of an event in *Day* is generalized to *Day of the Week* (Mon, Tue, etc.), then *Businessday/Weekend* (BD/WE), and finally *All_Time*(AT).

With the above settings, we conducted a series of experiments to evaluate our framework. In the following section, we present the results obtained from extracting location annotations for the five datasets. We then demonstrate the utility of these annotations in location clustering in Section 6.5.3. Finally, we show the efficiency of Algorithms 6.1 and 6.2 in Section 6.5.4.

6.5.2 Annotation Extraction

First, we run Algorithm 6.1 to obtain LT-Profiles for the five datasets for the two years (2011-2012). Then, annotations for locations are obtained using the method described in Section 6.4.3. Table 6.3 shows the result statistics, where the five datasets are run with different *npmi* thresholds (δ). Basically, the larger the threshold δ, the less locations are annotated, but the more confident the annotations are. With $\delta = 0.1$, about 70-90% of the locations were annotated, whereas less than 30% of the locations were annotated when $\delta > 0.5$.

Since there is no pre-existing ground-truth, we manually validate only the extracted annotations of the famous locations in the datasets by using external data sources (e.g., Google Search or Wikipedia). Table 6.4 shows annotations of some famous locations that we obtained. Note that the words describing topics are stemmed, and an item of a location annotation is followed by its *npmi* value, e.g., socc_WE:0.39 for '*soccer*' on Weekends.

By utilizing the annotations, one can easily find locations related to some given event topics. For example, *NürnbergMesse* (Germany) will be found when we search

[3]Event identifiers are used for two purposes: to distinguish an event from others and to link event contexts to tags.

Table 6.3: Result Statistics.

Dataset	**Total locations**	**The number of annotated locations**				
		$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.4$	$\delta = 0.5$
DE-Sports	960	808 (84%)	798 (83%)	769 (80%)	451 (46%)	331 (34%)
DE-Festival	1,515	1,163 (76%)	1,154 (76%)	988 (65%)	790 (52%)	539 (35%)
EU-Festival	18,018	16,237 (90%)	15,101 (83%)	13,636 (75%)	10,877 (60%)	7,144 (39%)
DE-Music	12,591	8,848 (70%)	7,385 (58%)	6,123 (48%)	4,811 (38%)	3,757 (29%)
DE-All	20,141	15,767 (78%)	12,097 (60%)	10,368 (51%)	7,561 (37%)	5,511 (27%)

Table 6.4: Annotations of some famous locations, extracted from the experimental datasets. Items in each annotation are sorted by their *npmi* values.

Location/Granularity	**Annotation**
	DE-Sports
Signal Iduna Park - Dortmund (Address)	{borussia_Sat:0.66, borussia_WE:0.61, borussia_AT:0.56, bundesliga_Sat:0.46, bundesliga_WE:0.42, football_WE:0.32, socc_WE:0.29,...}
Oschersleben Sachsen-Anhalt (City)	{circuitracing_AT:0.81, circuitracing_WE:0.77, motorsport_AT:0.71, autosport_AT:0.70, racing_AT:0.69, motorsport_WE:0.68,...}
	DE-Festival
Kino Babylon Mitte - Berlin (Address)	{filmfestival_BD:0.63, filmfestival_Thu:0.63, movi_BD:0.59, movi_Thu:0.59, film_BD:0.55, film_Thu:0.55, filmfestival_AT:0.54, movi_AT:0.50,...}
Messe Essen GmbH (Address)	{expo_Thu:0.66, fashion_Sat:0.66, convention_Thu:0.66, fashion_WE:0.64, homeexhibition_Fri:0.58, industry_AT:0.49, expo_BD:0.48,...}
	DE-Music
Bar/Night Club (Place category)	{elektronic_WE:0.55, hardstyl_WE:0.55, nightlif_WE:0.52, tranc_WE:0.45, rhythmnblu_BD:0.43, elektronic_AT:0.42, hardstyl_AT:0.42,...}
Concert Hall (Place category)	{philharmonieess_AT:0.67, doommetal_Tue:0.49, epic_Tue:0.49, jamsession_Mon:0.48, monstrosity_Wed:0.46, greatesthit_Mon:0.46,...}
	DE-ALL
Nürnberg Messe (Address)	{softwar_AT:0.62,expopromot_AT:0.53,school&alumni_AT:0.52, tool_AT:0.42, tradeshow_AT:0.41,scienc_AT:0.41, business_AT:0.41,...}
Philharmonie Berlin (Address)	{klassischekonzert_AT:0.64, cultur_AT:0.54, klassisch_AT:0.54, classical_AT:0.53, cultur_WE:0.49, symphony_WE:0.44, violin_Mon:0.42,...}
	EU-Festival
Torre del Lago - Tuscany - Italy (City)	{art&theatr_AT:0.87, art&theatr_BD:0.84, art&theatr_WE:0.73, opera_AT:0.27, opera_BD:0.26, opera_Fri:0.24, opera_WE:0.22,...}
LilianBaylisTheatre - London (Address)	{ballet_AT:0.80, ballet_BD:0.77, ballet_WE:0.69, clubbing_WE:0.47, nightlif_WE:0.47, danc_AT:0.40, theatr_Wed:0.32, art_Wed:0.28,...}

for places related to '*technology*' and '*exhibition*', as shown in Table 6.4. This can be explained by annual events related to computer software/hardware or electronic systems that are located here, such as '*embedded world*'. Similarly, one can discover that *Oschersleben*, a town in Sachsen-Anhalt, Germany, is strongly related to moto and auto sports.

From the extracted annotations, one can see that some annotations are obvious. For instance, the annotation of a cinema (e.g., *Kino Babylon Mitte - Berlin*) contains event topics related to film and movie festivals; the annotation of a football stadium in Germany (e.g., *Signal Iduna Park*) contains event topics related to '*soccer*' and '*bundesliga*'; or an exhibition centre (e.g., *Messe Essen GmbH*) contains event topics related to '*expo*', '*industry*', and '*tradeshow*'. Besides, we also found some interesting relationships, such as a relationship between the exhibition centre *Messe Essen GmbH* and the topic '*fashion*'. This relationship can be explained by a series of Modatex Fashion Fair events frequently occurring at that location.

We also discovered some cities in Europe that are famous for their annual festivals from the dataset EU-Festival. For instance, Torre del Lago, Peraso (Italy), and Montpelier (France) are famous for opera festivals.

At the *Place_Category* level, one can see many generic topics, for example, '*nightlife*' or '*electronic*' in *Bar/Night Club* extracted from the dataset DE-Music.

6.5.3 Location Clustering

We exploit the extracted annotations to cluster locations. Such clusters will be utilized further to assign higher level semantic tags to locations or to build taxonomies of locations, as described in Section 6.4.5. For this purpose, we employ hierarchical clustering. We tried with three common merging methods: Single-Link, Complete-Link, and Group-Average. The quality of a clustering solution is evaluated by a F-score measure, as commonly used in document clustering [63, 132]. Before detailing this F-score measure, we describe how to obtain datasets with ground-truth for clustering evaluation.

Since a location in our dataset can be generalized to a place category (e.g., '*Hotel*', '*Restaurant*'), we use such categories as ground-truth labels to evaluate location clustering, that is, locations of the same label are expected to be in the same cluster. However, our analysis shows that only 5% of the locations that are labeled with categorical tags might help for clustering evaluation. Other locations labeled as '*postal code*', '*address*', or '*named place*' produce meaningless results in the clustering eval-

Table 6.5: Top place categories for two datasets DE-All and EU-Festival.

DE-ALL		EU-Festival	
Category	**Number of locations**	**Category**	**Number of locations**
1. Bar/NightClub	174	1. Bar/NightClub	285
2. ConcertHall	79	2. ConcertHall	111
3. Stadium	71	3. Theatre	109
4. Theatre	33	4. Stadium	83
5. Restaurant	17	5. Park	62
6. Hotel	17	6. Museum	44
7. Museum	17	7. Outdoors_Field	43

uation method. To increase the number of locations with meaningful labels, we also utilized DBPedia[4], an LOD data source, to assign place categories for well-known locations, such as stadiums, hotels, or theatres. Based on simple text matching, we additionally assigned categorical tags (obtained from DBPedia) to about 100 locations (+25%) of the dataset DE-All and about 250 locations (+30%) of the dataset EU-Festival. Table 6.5 shows the top categories for the two datasets DE-All and EU-Festival in terms of the number of locations after performing this step. Using locations belonging to these categories, one can generate dataset with ground-truth for clustering evaluation.

We use $L^k_{\{C_1,C_2,...,C_k\}}$ to denote a dataset consisting of locations of k categories C_1, C_2,..., or C_k, e.g., $L^2_{\{Stadium,Theater\}}$. If one obtains k clusters $S_1, S_2, ..., S_k$ ($k \geq 2$) from a given dataset $L^k_{\{C_1,C_2,...,C_k\}}$, then the quality of this clustering solution is measured by the F-score [132]. In the following, we briefly describe how to compute this measure.

First, the Precision, Recall, and F-score for each pair of a category C_i and a cluster S_j are computed as

$$Pr(C_i, S_j) := \frac{n_{ij}}{n_j}, \quad Re(C_i, S_j) := \frac{n_{ij}}{n_i},$$

$$F(C_i, S_j) := \frac{2 * Pr(C_i, S_j) * Re(C_i, S_j)}{Pr(C_i, S_j) + Re(C_i, S_j)},$$

where n_i is the number of locations belonging to the category C_i, n_j is the size of the cluster S_j, and n_{ij} is the number of locations belonging to the category C_i in the cluster S_j. The F-score of each category C_i is then defined as the maximum F-score value for all clusters. That is,

$$F(C_i) := \max_{1 \leq j \leq k} F(C_i, S_j).$$

[4]We used the version 3.9 downloaded from http://dbpedia.org.

Finally, the F-score of the entire clustering solution for an input dataset $L^k_{\{C_1,C_2,...,C_k\}}$ is defined as

$$\textit{F-score}(L^k_{\{C_1,C_2,...,C_k\}}) := \sum_{i=1}^{k} \frac{n_i}{n} F(C_i), \tag{6.11}$$

where n_i is the number of locations belonging to the category C_i, and n is the number of locations in the input dataset.

As mentioned in Section 6.3.2, an alternative to *npmi* is *tf-idf* that can be employed to weight event topics for location clustering. Therefore, by employing the F-score described above, we now compare the performance of the *npmi* measure to the following versions of *tf-idf* that are widely used in document clustering [24]. Given an event topic f and an LTC Ω, two versions of *tf-idf*, denoted *tf-idf*$_1$ and *tf-idf*$_2$, are defined as

$$\textit{tf-idf}_1(f, \Omega) = N_{f,\Omega} * \log(\tfrac{N}{N_f}) \text{ and}$$

$$\textit{tf-idf}_2(f, \Omega) = (1 + \log(N_{f,\Omega})) * \log(\tfrac{N}{N_f}),$$

where the values N, N_f, and $N_{f,\Omega}$ are

- N : the number of LTCs,
- N_f : the number of LTCs that contain ET f,
- $N_{f,\Omega}$: the number of instances of the LTC Ω that support f.

Similar to the *npmi* measure, profiles of LTCs can be computed with Equation (6.6), where *npmi* is replaced by either *tf-idf*$_1$ or *tf-idf*$_2$. For a particular dataset and a particular measure, the threshold δ is selected so that the *F-score* in Formula (6.11) is the largest.

We use locations of the datasets DE-All and EU-Festival to assess the performance of location clustering since they cover all locations of the other datasets (i.e., DE-Sports, DE-Festival, and DE-Music). Using the place categories shown in Table 6.5, we generate datasets of locations for clustering evaluation. Instead of presenting the F-score for each generated dataset, we show the mean F-score for each group G_k of datasets containing the same number of categories (i.e., k categories). For example, with 7 categories as shown in Table 6.5, *F-score*(G_2) denotes the mean F-score for $\binom{7}{2} = 21$ datasets created by selecting 2 from the 7 categories, e.g., $L^2_{\{Stadium,Museum\}}$, $L^2_{\{Hotel,Museum\}}$, or $L^2_{\{Stadium,Theater\}}$.

Figure 6.6 shows the comparison of the *npmi* measure with *tf-idf*$_1$ and *tf-idf*$_2$ in location clustering. In general, using the *npmi* measure gives the best result. A closer

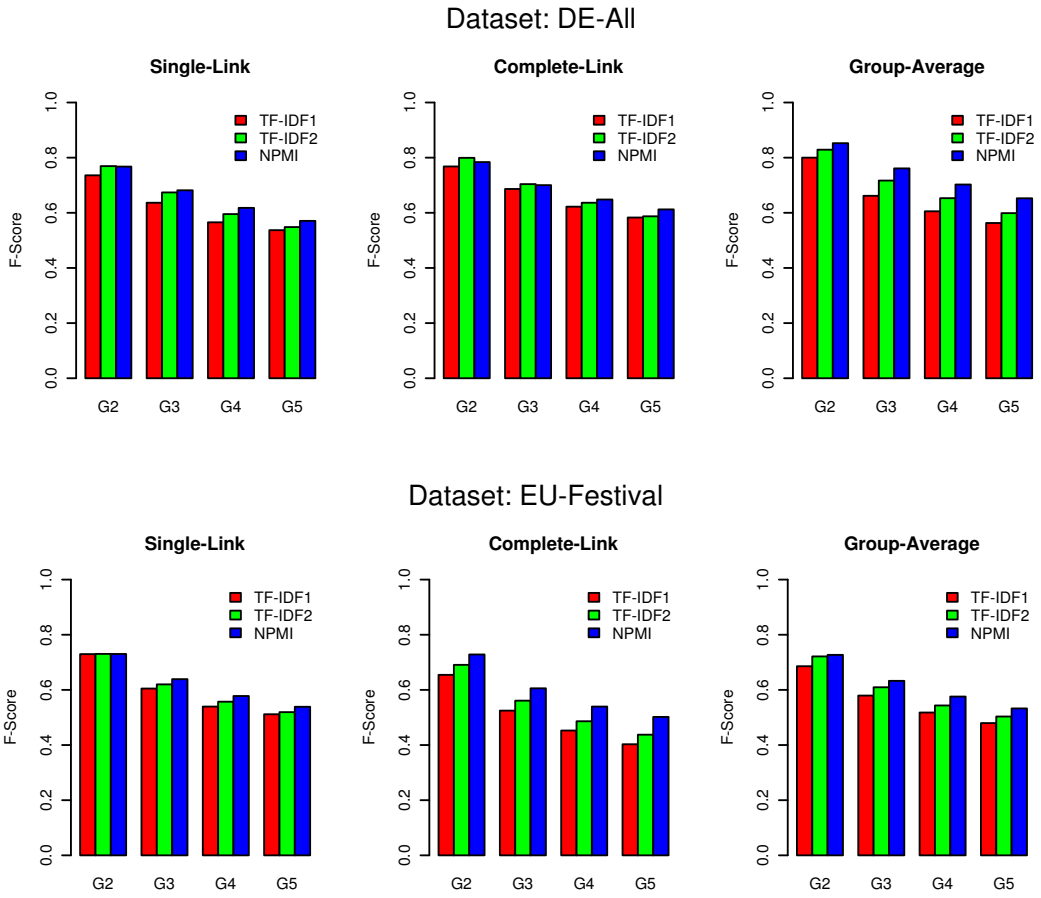


Figure 6.6: Comparison of the measures *tf-idf*$_1$, *tf-idf*$_2$, and *npmi* with various merging methods (Single-Link, Complete-Link, and Group-Average) of hierarchical clustering. The mean F-Score of each dataset group (G_2, G_3, G_4, and G_5) is shown for each measure. The best result is achieved with the *npmi* measure.

look at the generated profiles shows that a profile generated by the *npmi* measure contains more event topics presenting the characteristics of the corresponding location, as discussed in Section 6.3.2. Note that the main purpose of the *npmi* measure is to extract semantic annotations for locations. Although the *npmi* measure slightly outperforms the *tf-idf* measures, the above results indicate that one can indirectly evaluate how good the extracted annotations are by clustering. In addition, one can see from Figure 6.6 that clustering on locations of the dataset DE-All gives better results than clustering on locations of the dataset EU-Festival. This is reasonable since it is more difficult to categorize locations of the latter dataset consisting of narrow topics, i.e., event topics related to '*festival*'.

In Section 6.4.4, we introduced a dissimilarity distance for locations based on the Jaccard Index. Here, we compare the performance of this distance with the Cosine

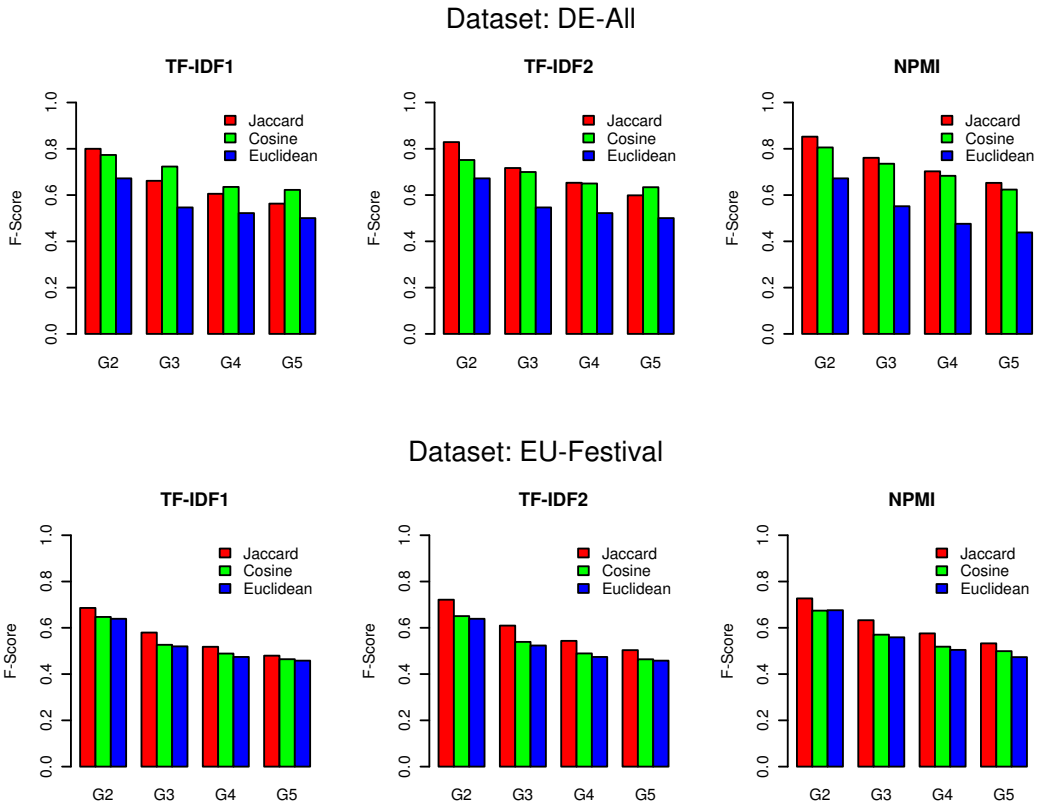


Figure 6.7: Comparison of the distance metrics *Jaccard*, *Cosine*, and *Euclidean* in hierarchical clustering with the Group-Average merging method. The mean F-Score of each dataset group (G_2, G_3, G_4, and G_5) is shown for each metric. The best result is achieved with the *Jaccard* distance.

distance and the Euclidean distance, two distance metrics commonly used in document clustering [111]. Figure 6.7 shows the result, where one can see that the Euclidean distance performs worst. This result agrees with several experimental results of text document clustering (e.g., [38, 107]). Generally, the Jaccard distance slightly outperforms the Cosine distance for the both datasets DE-All and EU-Festival.

Figure 6.8 shows an example of clustering based on location annotations extracted from the dataset DE-All. In general, locations in a cluster are of the same category. However, some locations are assigned to wrong clusters. For example, one can see that two locations of the category '*Bar/Club*' are misclassified as '*ConcertHall*'. After examining their annotations, we found that these locations have event topics related to music and performing art very similar to locations of the category '*ConcertHall*'.

For the other datasets, we also found many clusters that can be validated by using external sources (Google Search and Wikipedia). For example, we found clusters of bars/night clubs regarding their music genres (e.g., jazz or r&b/soul) such as the

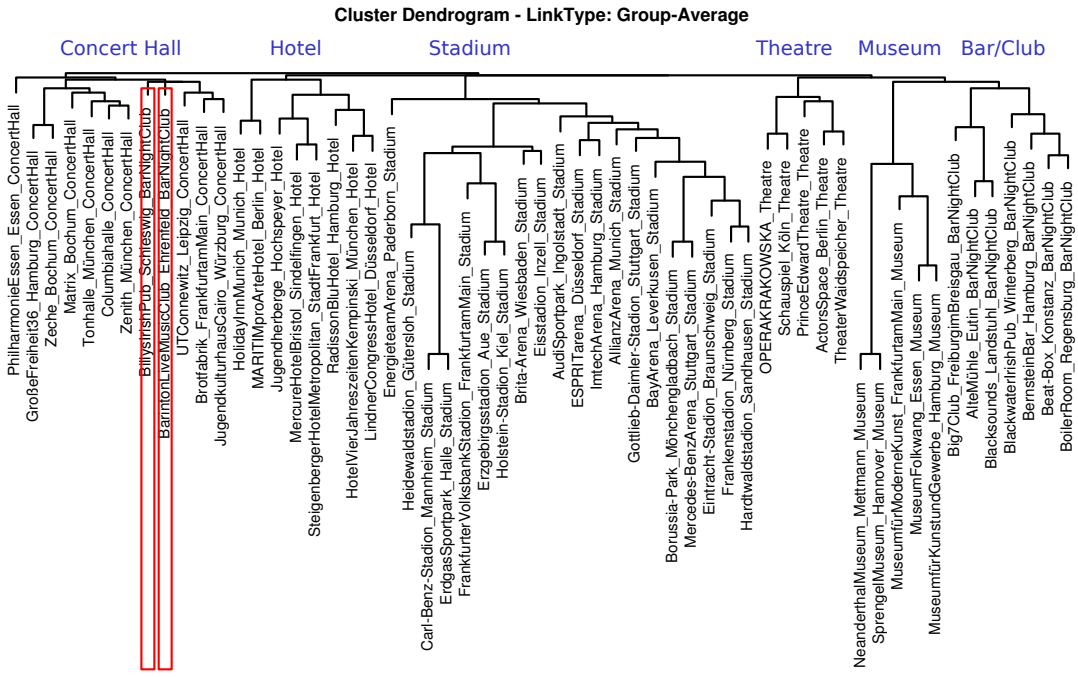


Figure 6.8: Example of clustering based on location annotations for the dataset DE-All. Locations are of granularity *Address*. Each location is shown with its place-category. Two locations (marked in boxes) of the category '*Bar/Club*' are in the same cluster as locations of the category '*ConcertHall*'.

Table 6.6: Example of bars/clubs (location instances) categorized by music genres of their events.

Jazz	**Rock & Metal**
Badehaus Szimpla Musiksalon - Friedrichshain	Hard Rock Berlin
Jazzclub Unterfahrt - München	Atomic Cafe - München
Hot Jazz Club - Münster	TNT Rock & Metal Pub - Niederkrüchten
Domkeller - Aachen	Kukuun Club - Hamburg
Mister B's München	Cafe Hahn - Koblenz
...	...
R&B/Soul	**Rap & Hiphop**
Löwensaal Stein bei Nürnberg	Tower Musikclub Bremen
Quasimodo - Berlin	LaViola Cafe & Weinbar - Siegburg
Live Club Barmen - Wuppertal	Discothek Halifax - Himmelkron
Engelsburg Bar/Night Club - Erfurt	Monofaktur - Muenchen
Little Stage - Kneipe Bühne Lounge Neukölln Live - Berlin	Club Metropolitain - Traunstein
...	...

four clusters presented in Table 6.6; or a cluster of cities in Europe famous for opera festivals like Montpellier (France), Torre del Lago (Italy), or Pesaro (Italy), as shown in Figure 6.9.

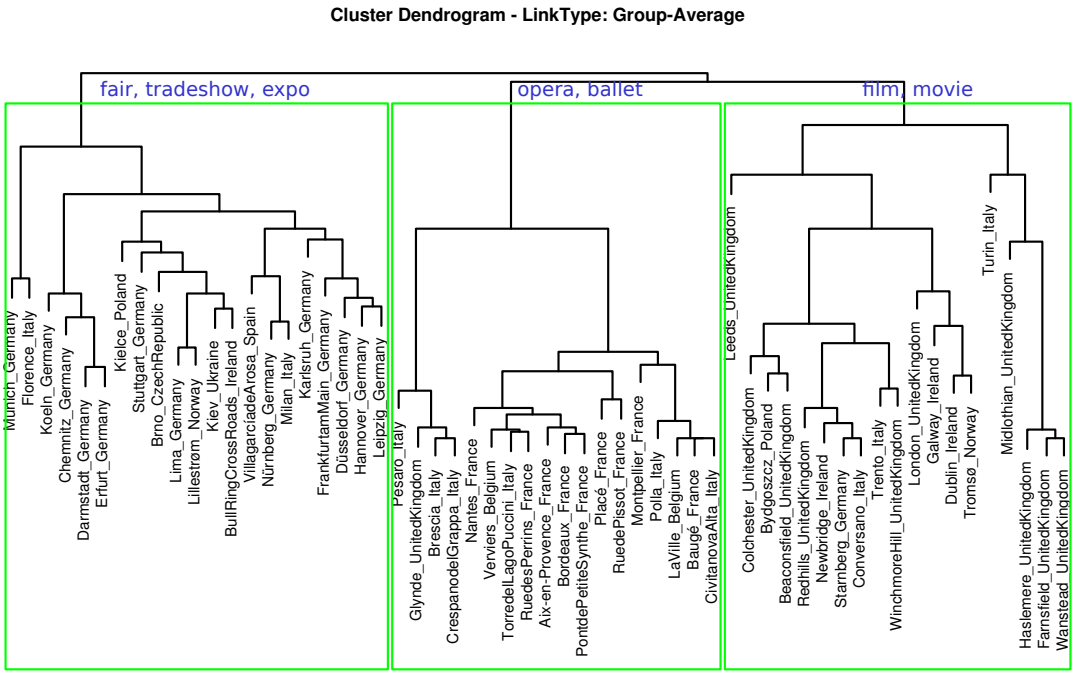


Figure 6.9: Example of clustering cities in the dataset EU-Festival based on their festival events. The left cluster contains cities that are related to event topics '*fair*', '*tradeshow*', and '*expo*'. The middle cluster contains cities that are related to event topics '*opera*' and '*ballet*'. The right cluster contains cities that are related to event topics '*film*' and '*movie*'.

6.5.4 Runtime and LTP-Updater Efficiency

In this section, we show the runtime of *LTProfile-Miner* for each dataset and also demonstrate the utility and efficiency of *LTProfile-Updater*. To do so, we split each dataset in Table 6.2 into two parts, each corresponding to one year. For example, from the dataset DE-Sports, we create two subdatasets DE-Sports$_{[2011]}$ and DE-Sports$_{[2012]}$, where the first one consists of events in 2011 and the latter one consists of events in 2012 of the dataset DE-Sports. From Table 6.2, one can see that the number of events in 2012 is about 20 to 50% larger than in 2011.

For each triple of datasets ($\mathcal{D}$, $\mathcal{D}_{[2011]}$ and $\mathcal{D}_{[2012]}$), we measure the runtime t_1 for *LTProfile-Miner* on $\mathcal{D}_{[2011]}$, the runtime t_2 for *LTProfile-Miner* on $\mathcal{D}$, and the runtime t_3 for *LTProfile-Updater* on $\mathcal{D}_{[2012]}$ (with support-data extracted from $\mathcal{D}_{[2011]}$). Figure 6.10 shows such runtimes for each dataset, where the datasets are sorted by the number of events. The first two datasets take only a few seconds to process. The cases of EU-Festival and DE-Music illustrate that the number of LTIs also affects the runtime, as mentioned in the complexity analysis of Algorithm 6.1. Although the number of events of the dataset EU-Festival is smaller than the dataset DE-Music,

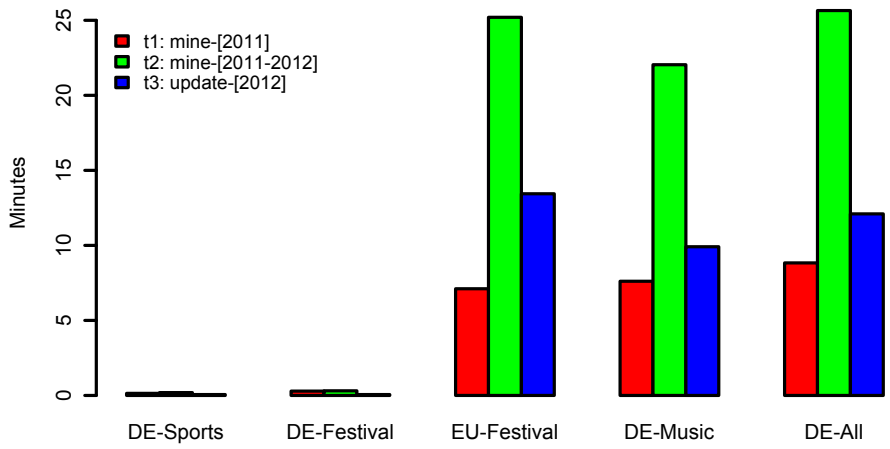


Figure 6.10: Runtime of LTP-Miner and LTP-Updater. For each dataset, t_1 is the runtime to mine events in 2011, t_2 is the runtime to mine events in [2011-2012], and t_3 is the runtime to update with events in 2012.

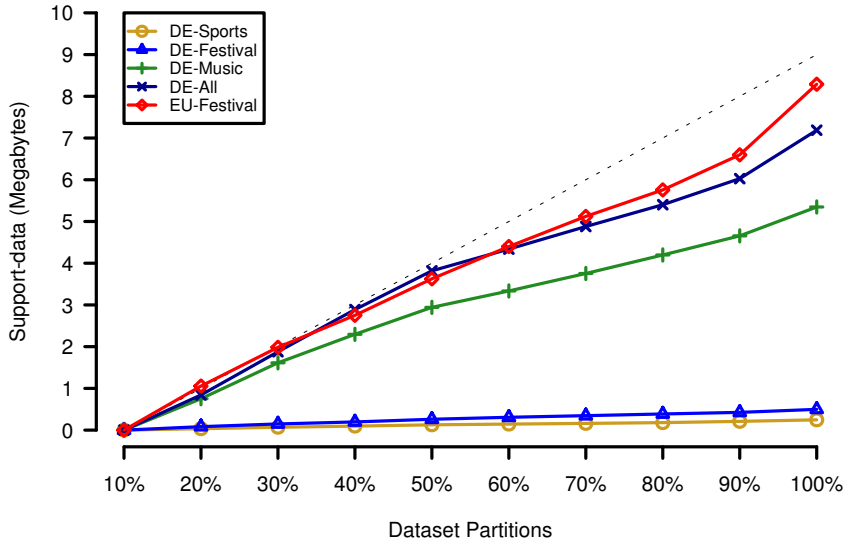


Figure 6.11: Increase of support-data size after a step of updating 10% of a dataset.

the number of locations of the dataset EU-Festival is larger, as shown in Table 6.2. In all cases, the runtime t_3 is larger than t_1, because the number of events in 2012 is larger than in 2011. However, in comparison to t_2, the runtime t_3 is much smaller for both datasets. This shows that using the *LTProfile-Updater* is an efficient and scalable approach to update the current location profiles with new data.

We finally conducted several experiments related to the size of the support-data. Since the support-data is sparse, we use a data structure that employs a hash-table

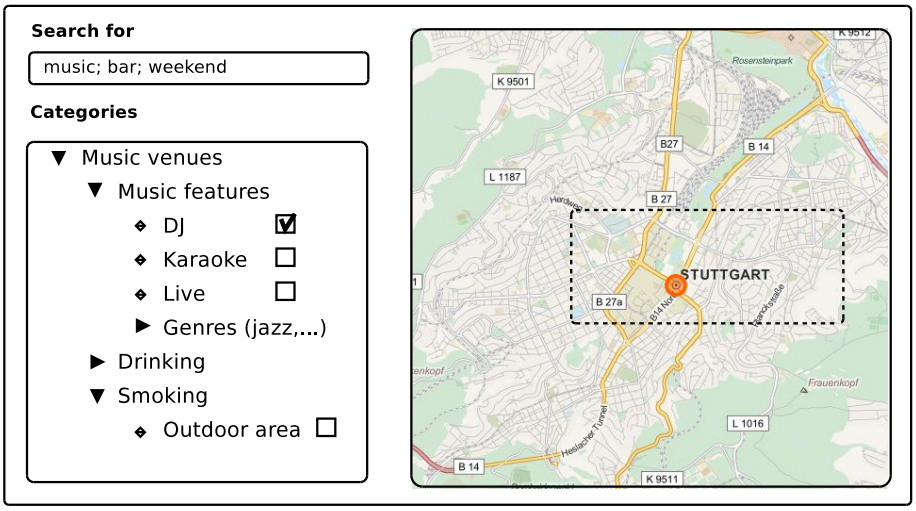


Figure 6.12: Example of a graphic user interface that enables the user to refine the result of searching for locations. The user can select topics of interest from a taxonomy that is automatically generated based on the input query, and/or specify a bounding box on the map to constrain the search space.

to save space and also to efficiently access the data. For each dataset in Table 6.2, we sort the events by time and equally divide them into 10 partitions. We run *LTProfile-Miner* for the first partition and *LTProfile-Updater* for the others. After each step, we store the support-data to files for the next update. Figure 6.11 shows the size of support-data after each step. One can see that the size of support-data is linear with respect to the size of the dataset. Additionally, the support-data size depends on how sparse its content is. Thus, for a dataset related to a specific topic (like EU-Festival), the content of its support-data tends to be denser than for a dataset consisting of various topics (like DE-All). Thus, as one can see in the Figure 6.11, the size of the support-data for EU-Festival is slightly larger than for DE-All.

6.6 Discussion

Event-based annotations of locations describe the event topics that are most related to a location, together with the time when the events of such topics most likely occur. In this chapter, we presented a comprehensive framework to extract such annotations from event datasets. We also proposed a scalable and efficient method to deal with periodic updates of event data. Our experimental results clearly indicated that the extracted annotations can be utilized for semantic location search as well as clustering.

The above framework is clearly fundamental to develop a location-based service application or a recommender system. For example, Figure 6.12 illustrates a graphic user interface that utilizes location annotations to guide the user to perform semantic search for locations. Based on keywords (e.g., '*music*', '*bar*', '*weekend*') specified by the user, a taxonomy of relevant topics is automatically generated for the user to refine the list of locations. The user might also specify a bounding box on the map to constrain the geographic search space. In such a system, a database that manages information about locations plays a very important role in the back end. Our framework provides fundamental components to create and incrementally update that database by exploiting and monitoring some external data source of events.

Chapter 7

Conclusions and Future Work

This chapter summarizes the thesis and provides a discussion of open research problems for future work.

7.1 Summary and Conclusions

Knowledge and patterns derived from event data are clearly valuable in providing a semantically rich basis for real-world applications such as location-based services or recommender systems. In this thesis, we presented novel approaches to the discovery of interesting, useful patterns from event data. In particular, we built a comprehensive framework to model events, where each event is described by three components: context, time, and location. Accordingly, conceptual, temporal, and spatial (geographic) relationships between events can flexibly be formulated, and various constraints on events can naturally be specified. Moreover, the three components of an event can be individually generalized to higher levels of abstraction to derive event templates, which represent topics of events and play an important role in building event patterns. Aiming at a flexible framework to model events, the notations of events and event templates as well as different relationships among them are very generally defined. However, they can easily be specialized for a particular pattern mining approach, as demonstrated in Chapters 4, 5, and 6.

By employing the above framework, we addressed the research problem of mining interval-based event sequence patterns (IESPs) from event data. This problem can be considered a generalization of mining sequential patterns from traditional data (i.e., sequences of items or sequences of transactions) for the following reasons. First, our pattern language is able to express not only time point-based but also interval-based relationships between events. Second, supporting multiple temporal predicates, e.g.,

before, *after*, or *overlaps*, in a single pattern enables the pattern language to express complex temporal relationships among events. As a type of spatio-temporal patterns, IESPs also capture spatio-temporal relationships between events. Furthermore, our approach utilizes concept hierarchies associated with event components to find IESPs at different levels of abstraction and granularity. As shown in the experiments, beside obvious patterns, we also obtained patterns that describe interesting and meaningful relationships among events from real datasets.

The second research problem that we focused on is to find periodic event patterns from event data. In particular, we formulated a notion of relaxed periodicity for single event topics as well as for sets of event topics that frequently occur together. For this, we proposed a probabilistic measure to determine how likely an event topic occurs or multiple event topics co-occur during a time window. In addition, our approach can find interesting periodic patterns at different levels of abstraction and granularity, as demonstrated with real event datasets.

In the last approach presented in this thesis, we demonstrated that not only correlations among events but also correlations among event components can be exploited to extract valuable knowledge. For this, we presented a comprehensive framework to extract semantic annotations for locations. Our framework provides the basis to efficiently derive characteristic, discriminative information for locations, and make this information up-to-date when new events are added to the current dataset. We demonstrated the utility and efficiency of the framework with real event datasets.

In conclusion, we showed that data sources of events provided by social media channels can be well exploited for valuable knowledge. Our experiments clearly indicated that extracted patterns and knowledge can be well utilized in various practically relevant tasks, such as event prediction, semantic search for locations, or topic-based location clustering. Furthermore, this study on mining patterns from event data provides a foundation for several interesting, promising directions of future work, as discussed next.

7.2 Future Work

Based on the results presented in this thesis, several promising directions of future work can be conducted. They can be categorized into the following aspects:

- **Extensibility**: Generally, event relationships are fundamental to formulate patterns of events. As shown through this thesis, we derived event relationships

on the basis of spatial and temporal proximities as well as of topic-based similarity of the corresponding events. Besides these familiar relationships, additional semantic relationships, such as *part-whole* or *meronymic* relationships [106], between events might be explicitly provided by some data sources of events. For example, a workshop session is a part of a data mining conference; a festival often consists of multiple sub-events; or a disease might consist of several phases. The existence of such relationships between events in a dataset gives rise to new approaches to the discovery of more complex types of patterns, such as tree-like or graph-like patterns, of events. Therefore, integrating semantic relationships as mentioned above into the event model is a promising direction of future work to fully exploit that type of event data.

Another direction that might be an interesting and promising investigation is to adapt and extend the approaches described in this thesis to find patterns from event data of other domains, such as for scientific data. For example, in the context of weather and climate event data, *spatio-temporal teleconnection patterns* [60] are commonly known as connections between climate phenomena of two distant regions that are correlated with each other. Such patterns can be discovered at different levels of abstraction and granularity by employing our event framework. However, further research needs to be conducted, for example, to transform sensor observation data to events, to generate concept hierarchies for these events, and to design a good measure for interesting patterns.

- **Applicability**: Exploiting event patterns for other data mining tasks, such as pattern-based classification, outlier detection, or event prediction, is an interesting direction of future work. For example, information about sequential and periodic patterns derived from an event dataset might be valuable for predicting events or for identifying anomalies in similar datasets of events. Location annotations as described in Chapter 6 can be exploited to automatically build taxonomies of locations. Furthermore, utilizing event patterns to real-life problems such as location-based services or recommender systems still needs to be investigated.

- **Scalability**: In reality, new events are periodically inserted into the current database of events. One therefore needs a scalable, efficient approach to make discovered patterns up-to-date instead of re-performing the whole process for both old and new events. For this, we already proposed an incremental approach in Chapter 6 to update annotations of locations. Recall that we formu-

lated annotations of locations on the basic of correlations among components of events. The key point of the approach is that significant correlations among event components can be individually computed and then aggregated from the set of existing events and the set of only new events. For other types of event patterns that are formulated by exploiting correlations among events (not event components), partitioning event data like that might not work since new relationships might appear not only between two new events but also between a new event and an existing one. Therefore, investigating an incremental approach for these event patterns is still an open problem.

Another obvious direction is to parallelize the process of pattern mining to support large-scale, distributed databases of events. Employing an existing parallelization framework, such as the MapReduce [28], might be an efficient solution for such large-scale data. However, creating and managing parallel tasks in such an approach are not trivial since the pattern pruning that performs on a task requires the previous pattern computations of some other tasks. Thus, efficiently sharing that information across parallel tasks is necessary and still an open problem.

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB*, pages 487–499. Morgan Kaufmann, 1994.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Mining Sequential Patterns. In *ICDE*, pages 3–14. IEEE Computer Society, 1995.

[3] James Allan and Ron Papka. On-line New Event Detection. In *SIGIR*, pages 37–45. ACM, 1998.

[4] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26:832–843, 1983.

[5] Luis Otavio Alvares, Vania Bogorny, Bart Kuijpers, Jose Antonio Fernandes de Macedo, Bart Moelans, and Alejandro Vaisman. A Model for Enriching Trajectories with Semantic Geographical Information. In *ACM GIS*, pages 22:1–22:8. ACM, 2007.

[6] Pierre Andrews, Ilya Zaihrayeu, and Juan Pane. A Classification of Semantic Annotation Systems. *Semantic Web*, 3(3):223–248, 2011.

[7] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *Semantic Web Conference*, pages 722–735. Springer-Verlag, 2007.

[8] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential Pattern Mining Using a Bitmap Representation. In *KDD*, pages 429–435. ACM, 2002.

[9] Lamberto Ballan, Marco Bertini, Alberto Bimbo, Lorenzo Seidenari, and Giuseppe Serra. Event Detection and Recognition for Semantic Annotation of Video. *Multimedia Tools and Applications*, 51(1):279–302, 2010.

[10] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(2):1–22, 2009.

[11] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A Crystallization Point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.

[12] J.A. Bondy and U.S.R. Murty. *Graph Theory*, volume 244 of *Graduate texts in mathematics*. Springer, 2008.

[13] Gerlof Bouma. Normalized (Pointwise) Mutual Information in Collocation Extraction. In *From Form to Meaning: Processing Texts Automatically, Proceedings of the Biennial GSCL Conference*, pages 31–40, 2009.

[14] Coen Bron and Joep Kerbosch. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM*, 16(9):575–577, 1973.

[15] Alex G. Büchner and Maurice D. Mulvenna. Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining. *ACM SIGMOD Record*, 27(4):54–61, December 1998.

[16] Huiping Cao, N. Mamoulis, and D.W. Cheung. Mining Frequent Spatio-Temporal Sequential Patterns. In *ICDM*, pages 82–89. IEEE Computer Society, 2005.

[17] Huiping Cao, Nikos Mamoulis, and David Cheung. Discovery of Collocation Episodes in Spatiotemporal Data. In *ICDM*, pages 823–827. IEEE Computer Society, 2006.

[18] Huiping Cao, Nikos Mamoulis, and David Cheung. Discovery of Periodic Patterns in Spatiotemporal Sequences. *TKDE*, 19(4):453–467, 2007.

[19] Xin Cao, Gao Cong, and Christian S Jensen. Mining Significant Semantic Locations From GPS Data. *Proceedings of the VLDB Endowment*, 3(1-2):1009–1020, 2010.

[20] Mete Celik, Shashi Shekhar, James P. Rogers, and James a. Shine. Mixed-Drove Spatiotemporal Co-Occurrence Pattern Mining. *TKDE*, 20(10):1322–1335, 2008.

[21] Dipanjan Chakraborty, Stefano Spaccapietra, and Christine Parent. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. In *EDBT*, pages 259–270, 2011.

[22] Y Chen and Y Hu. Constraint-based Sequential Pattern Mining: the Consideration of Recency and Compactness. *Decision Support Systems*, 42(2):1203–1215, 2006.

[23] ChongWang, David Blei, and Li Fei-Fei. Simultaneous Image Classification and Annotation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1903–1910, 2009.

[24] Christopher D. Manning, Raghavan Prabhakar, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2009.

[25] Kenneth Ward Church and Patrick Hanks. Word Association Norms, Mutual Information, and Lexicography. *Comput. Linguist.*, 16(1):22–29, 1990.

[26] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Artificial Intelligence Research*, 24:305–339, 2005.

[27] N. Cressie. Statistics for Spatial Data. *Wiley Series in Probability and Mathematical Statistics*, 4:613–617, 1992.

[28] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[29] Leon R. a. Derczynski, Bin Yang, and Christian S. Jensen. Towards Context-aware Search and Analysis on Social Media Data. In *EDBT*, pages 137–142. ACM, 2013.

[30] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. GRAMI: Frequent Subgraph and Pattern Mining in a Single Large Graph. *Proceeding of the VLDB Endowment*, 7(7):517–528, 2014.

[31] Martin Erwig, Ralf Hartmut Güting, Markus Schneider, and Michalis Vazirgiannis. Abstract and Discrete Modeling of Spatio-temporal Data Types. In *ACM GIS*, pages 131–136. ACM, 1998.

[32] Martin Erwig and Markus Schneider. Spatio-temporal Predicates. *TKDE*, 14(4):881–901, 2002.

[33] Vladimir Estivill-Castrol and Ickjai Lee. Data Mining Techniques for Autonomous Exploration of Large Volumes of Geo-referenced Crime Data. In *International Conference on Geocomputation*. GeoCompuatation, 2001.

[34] Vladimir Estivill-Castrol and Alan Murray. Discovering Associations in Spatial Data - An Efficient Medoid Based Approach. In *PAKDD*, pages 110–121. Springer-Verlag, 1998.

[35] Jonathan G Fiscus and George R Doddington. Topic Detection and Tracking Evaluation Overview. In *Topic detection and tracking*, pages 17–31. Kluwer Academic Publishers, 2002.

[36] Minos N Garofalakis. SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *VLDB*, pages 223–234. Morgan Kaufmann, 1999.

[37] Arthur Getis and J. Keith Ord. The Analysis of Spatial Association by Use of Distance Statistics. *Geographical Analysis*, 24(3):189–206, 1992.

[38] J Ghosh and A Strehl. Similarity-Based Text Clustering: A Comparative Study. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data*, pages 73–97. Springer Berlin Heidelberg, 2006.

[39] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting Influenza Epidemics Using Search Engine Query Data. *Nature*, 457(7232):1012–4, February 2009.

[40] Ralf Hartmut Güting. An Introduction to Spatial Database Systems. *The VLDB Journal*, 3(4):357–399, October 1994.

[41] Jiawei Han and Yongjian Fu. Discovery of Multiple-Level Association Rules from Large Databases. In *VLDB*, pages 420–431. Morgan Kaufmann, 1995.

[42] Jiawei Han, Wan Gong, and Yiwen Yin. Mining Segment-Wise Periodic Patterns in Time-Related Databases. In *KDD*, pages 214–218. ACM, 1998.

[43] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, 3rd ed.* Morgan Kaufmann, 2011.

[44] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. FreeSpan: Frequent Pattern-projected Sequential Pattern Mining. In *KDD*, pages 355–359. ACM, 2000.

[45] Jiawei Han and Yiwen Yin. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *ICDE*, pages 106–115. IEEE Computer Society, 1999.

[46] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 2011.

[47] Vinod Hegde, Josiane Xavier Parreira, and Manfred Hauswirth. Semantic Tagging of Places Based on User Interest Profiles from Online Social Networks. In *ECIR*, pages 218–229. Springer-Verlag, 2013.

[48] Johannes Hoffart, Edwin L-Kelham, Fabian.M Suchanek, Gerard de Melo, Klaus Berberich, and Gerhard Weikum. YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. In *WWW*, pages 229–232. ACM, 2011.

[49] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Research Report MPI-I-2010-5-007, Max-Planck-Institut für Informatik, 2010.

[50] Frank Höppner. Discovery of Temporal Patterns. Learning Rules about the Qualitative Behaviour of Time Series. In *PKDD*, pages 192–203. Springer-Verlag, 2001.

[51] Y. Huang, S. Shekhar, and H. Xiong. Discovering Colocation Patterns from Spatial Data Sets: A General Approach. *TKDE*, 16(12):1472–1485, 2004.

[52] Yan Huang, Jian Pei, and Hui Xiong. Mining Co-Location Patterns with Rare Events from Spatial Data Sets. *GeoInformatica*, 10(3):239–260, 2006.

[53] Yan Huang, Liqin Zhang, and Pusheng Zhang. A Framework for Mining Sequential Patterns from Spatio-Temporal Event Data Sets. *TKDE*, 20(4):433–448, 2008.

[54] Yan Huang and Pusheng Zhang. On the Relationships between Clustering and Spatial Co-location Pattern Mining. In *IEEE International Conference on Tools with Artificial Intelligence*, pages 513–522. IEEE Computer Society, 2006.

[55] Piotr Indyk, Nick Koudas, and S. Muthukrishnan. Identifying Representative Trends in Massive Time Series Data Sets Using Sketches. In *VLDB*, pages 363–372. Morgan Kaufmann, 2000.

[56] Jay J Jiang and David W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *International Conference on Research in Computational Linguistics*, pages 19–33, 1997.

[57] Tao Jiang and Ah-Hwee Tan. Mining RDF Metadata for Generalized Association Rules. In *DEXA*, number 4080 in LNCS, pages 223–233. Springer-Verlag, 2006.

[58] Tao Jiang, Ah-hwee Tan, and Ke Wang. Mining Generalized Associations of Semantic Relations from Textual Web Content. *TKDE*, 19(2):164–179, 2007.

[59] Po-shan Kam and Ada Wai-Chee Fu. Discovering Temporal Patterns for Interval-Based Events. In *DaWaK*, pages 317–326. Springer-Verlag, 2000.

[60] Jaya Kawale, Snigdhansu Chatterjee, Dominick Ormsby, Karsten Steinhaeuser, Stefan Liess, and Vipin Kumar. Testing the Significance of Spatio-temporal Teleconnection Patterns. In *KDD*, pages 642–651. ACM, 2012.

[61] Krzysztof Koperski and Jiawei Han. Discovery of Spatial Association Rules in Geographic Information Databases. In *Proceedings of the 4th International Symposium on Advances in Spatial Databases*, pages 47–66. Springer-Verlag, 1995.

[62] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective Annotation of Wikipedia Entities in Web Text. In *KDD*, pages 457–466. ACM, 2009.

[63] Bjornar Larsen and Chinatsu Aone. Fast and Effective Text Mining Using Linear-time Document Clustering. In *KDD*, pages 16–22. ACM, 1999.

[64] Hady W Lauw, Ee-peng Lim, and Teck-tim Tan. Mining Social Network from Spatio-Temporal Events. *Computational and Mathematical Organization Theory*, 11(2):97–118, 2005.

[65] Anh Le and Michael Gertz. Mining Spatio-temporal Patterns in the Presence of Concept Hierarchies. In *ICDM Workshops*, pages 765–772. IEEE Computer Society, 2012.

[66] Anh Le and Michael Gertz. Mining Periodic Event Patterns from RDF Datasets. In *ADBIS*, volume 8133 of *Lecture Notes in Computer Science*, pages 162–175. Springer-Verlag, 2013.

[67] Anh Le, Michael Gertz, and Christian Sengstock. An Event-based Framework for the Semantic Annotation of Locations. In *ADBIS*, Lecture Notes in Computer Science. Springer-Verlag, 2014.

[68] C. Leacock and M. Chodorow. *Combining Local Context and WordNet Similarity for Word Sense Identification*, pages 305–332. MIT Press, 1998.

[69] Fritz Lehmann. Big Posets of Participatings and Thematic Roles. In *Proceedings of the 4th International Conference on Conceptual Structures: Knowledge Representation as Interlingua*, pages 50–74. Springer-Verlag, 1996.

[70] Michael Lesk. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM, 1987.

[71] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. Mining Periodic Behaviors for Moving Objects. In *KDD*, pages 1099–1108. ACM, 2010.

[72] Zhenhui Li, Cindy Xide Lin, Bolin Ding, and Jiawei Han. Mining Significant Time Intervals for Relationship Detection. In *SSTD*, pages 386–403. Springer-Verlag, 2011.

[73] Zhenhui Li, Jingjing Wang, and Jiawei Han. Mining Event Periodicity from Incomplete Observations. In *KDD*, pages 444–452. ACM, 2012.

[74] N.R. Lomb. Least-squares Frequency Analysis of Unequally Spaced Data. *Astrophysics and Space Science*, 39:447–462, 1976.

[75] Sheng Ma and J.L. Hellerstein. Mining Partially Periodic Event Patterns with Unknown Periods. In *ICDE*, pages 205–214. IEEE Computer Society, 2001.

[76] Nizar R. Mabroukeh and C. I. Ezeife. A Taxonomy of Sequential Pattern Mining Algorithms. *ACM Computing Surveys*, 43(1):1–41, November 2010.

[77] Nikos Mamoulis, Huiping Cao, George Kollios, Marios Hadjieleftheriou, Yufei Tao, and David W. Cheung. Mining, Indexing, and Querying Historical Spatiotemporal Data. In *KDD*, pages 236–245. ACM, 2004.

[78] Heikki Mannila and Hannu Toivonen. Levelwise Search and Borders of Theories in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[79] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[80] Eric Margolis and Stephen Laurence. Concepts. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab - Center for the Study of Language and Information - Stanford University, fall 2012 edition, 2012.

[81] Harvey J. Miller and Jiawei Han, editors. *Geographic Data Mining and Knowledge Discovery, Second Edition*. CRC Press, 2009.

[82] Fabian Mörchen. Unsupervised Pattern Mining from Symbolic Temporal Data. *ACM SIGKDD Explorations Newsletter*, 9(1):41–55, June 2007.

[83] Yasuhiko Morimoto. Mining Frequent Neighboring Class Sets in Spatial Databases. In *KDD*, pages 353–358. ACM, 2001.

[84] D.O. Olguin, B.N. Waber, Taemie Kim, A. Mohan, K. Ara, and A. Pentland. Sensible Organizations: Technology and Methodology for Automatically Measuring Organizational Behavior. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):43–55, February 2009.

[85] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic Association Rules. In *ICDE*, pages 412–421. IEEE Computer Society, 1998.

[86] Patrick Pantel, Dekang Lin, and Alberta T H Canada. Discovering Word Senses from Text. In *KDD*, pages 613–619. ACM, 2002.

[87] Dhaval Patel, Wynne Hsu, and Mong Li Lee. Mining Relationships Among Interval-based Events for Classification. In *SIGMOD*, pages 393–404. ACM, 2008.

[88] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Demonstration Papers at HLT-NAACL'04*, pages 38–44, 2004.

[89] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *ICDE*, pages 215–224. IEEE Computer Society, 2001.

[90] Jian Pei, Jiawei Han, and Wei Wang. Constraint-based Sequential Pattern Mining: the Pattern-growth Methods. *Intelligent Information Systems*, 28(2):133–160, January 2005.

[91] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Quiming Chen, and Umeshwar Dayal. Multi-dimensional Sequential Pattern Mining. In *CKIM*, pages 81–88. ACM, 2001.

[92] Marc Plantevit, Anne Laurent, Dominique Laurent, Maguelonne Teisseire, and Yeow Wei Choong. Mining Multidimensional and Multilevel Sequential Patterns. *ACM TKDD*, 4(1):1–37, January 2010.

[93] Daniele Quercia, Neal Lathia, Francesco Calabrese, Giusy Di Lorenzo, and Jon Crowcroft. Recommending Social Events from Mobile Phone Location Data. In *ICDM*, pages 971–976. IEEE Computer Society, 2010.

[94] Tye Rattenbury, Nathaniel Good, and Mor Naaman. Towards Automatic Extraction of Event and Place Semantics from Flickr Tags. In *SIGIR*, pages 103–110. ACM, 2007.

[95] Tye Rattenbury and Mor Naaman. Methods for Extracting Place Semantics from Flickr Tags. *ACM Transactions on the Web*, 3(1):1–30, January 2009.

[96] Philip Resnik. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[97] John F. Roddick and Myra Spiliopoulou. A Bibliography of Temporal, Spatial and Spatio-temporal Data Mining Research. *ACM SIGKDD Explorations Newsletter*, 1(1):34–38, June 1999.

[98] Mark Sanderson and Bruce Croft. Deriving Concept Hierarchies from Text. In *SIGIR*, pages 206–213. ACM, 1999.

[99] Jeffrey D Scargle, Jay Norris, and Brad Jackson. Statistical Aspects of Spectral Analysis of Unevenly Spaced Data. *Astrophysical Journal*, 1982.

[100] Christian Sengstock and Michael Gertz. Latent Geographic Feature Extraction from Social Media. In *SIGSPATIAL*, pages 149–158. ACM, 2012.

[101] Ryan Shaw and Lynda Hardman. LODE: Linking Open Descriptions of Events. In *Proceedings of the 4th Asian Conference on The Semantic Web*, pages 153–167. Springer-Verlag, 2009.

[102] Ru Shen, Nalin C W Goonesekere, and Chittibabu Guda. Mining Functional Subgraphs from Cancer Protein-protein Interaction Networks. *BMC systems biology*, 6(3):1–14, January 2012.

[103] Ramakrishnan Srikant and Rakesh Agrawal. Mining Generalized Association Rules. In *VLDB*, pages 407–419. Morgan Kaufmann, 1995.

[104] Ramakrishnan Srikant and Rakesh Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *EDBT*, pages 3–17. Springer-Verlag, 1996.

[105] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. LinkedGeoData: A Core for a Web of Spatial Open Data. *Semantic Web Journal*, 3(4):333–354, 2012.

[106] Vede C. Storey. Understanding Semantic Relationships. *The VLDB Journal*, 2(4):455–488, October 1993.

[107] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Measures on Web-page Clustering. In *Workshop on Artificial Intelligence for Web Search*, pages 58–64, 2000.

[108] Etsuji Tomita and Tomokazu Seki. An Efficient Branch-and-bound Algorithm for Finding a Maximum Clique. In *Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science*, pages 278–289. Springer-Verlag, 2003.

[109] Raphaël Troncy, Bartosz Malocha, and André T. S. Fialho. Linking Events with Media. In *Proceedings of the 6th International Conference on Semantic Systems*, pages 1–4. ACM, 2010.

[110] Ilias Tsoukatos and Dimitrios Gunopulos. Efficient Mining of Spatiotemporal Patterns. In *SSTD*, pages 425–442. Springer-Verlag, 2001.

[111] Peter D Turney and Patrick Pantel. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, January 2010.

[112] Willem Robert van Hage, Véronique Malaisé, Roxane Segers, Laura Hollink, and Guus Schreiber. Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2):128–136, July 2011.

[113] Charles Van Loan. *Computational Frameworks for the Fast Fourier Transform*. Society for Industrial and Applied Mathematics, 1992.

[114] Michail Vlachos, Philip Yu, and Vittorio Castelli. On Periodicity Detection and Structural Periodic Similarity. In *SDM*, pages 449–460. SIAM, 2005.

[115] Junmei Wang, Wynne Hsu, and Mong Li Lee. A Framework for Mining Topological Patterns in Spatio-temporal Databases. In *CIKM*, pages 429–436. ACM, 2005.

[116] Ke Wang, Yabo Xu, and Jeffrey Xu Yu. Scalable Sequential Pattern Mining for Biological Sequences. In *CIKM*, pages 178–187. ACM, 2004.

[117] Ting Wang, Mudhakar Srivatsa, Dashi Agrawal, and Ling Liu. Spatio-temporal Patterns in Network Events. In *Co-NEXT*, volume 5, pages 3:1–3:12. ACM, 2010.

[118] W. Wang and J. Yang. *Mining Sequential Patterns from Large Data Sets*. Advances in Database Systems. Springer, 2006.

[119] Michael F. Worboys. A Generic Model for Planar Geographical Objects. *International journal of geographical information systems*, 6(5):353–372, September 1992.

[120] Shin-Yi Wu and Yen-Liang Chen. Mining Nonambiguous Temporal Patterns for Interval-Based Events. *TKDE*, 19:742–758, 2007.

[121] Zhibiao Wu and Martha Palmer. Verbs Semantics and Lexical Selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[122] Xiangye Xiao, Xing Xie, Qiong Luo, and Wei-Ying Ma. Density Based Co-location Pattern Discovery. In *SIGSPATIAL*, pages 29:1–29:10. ACM, 2008.

[123] X. Yan and Jiawei Han. gSpan: Graph-Based Substructure Pattern Mining. In *ICDM*, volume 3, pages 721–724. IEEE Computer Society, 2002.

[124] Hui Yang, Srinivasan Parthasarathy, and Sameep Mehta. A Generalized Framework for Mining Spatio-temporal Patterns in Scientific Data. In *KDD*, pages 716–721. ACM, 2005.

[125] Jiong Yang, Wei Wang, and Philip S Yu. InfoMiner: Mining Surprising Periodic Patterns. In *KDD*, pages 395–400. ACM, 2001.

[126] Jiong Yang, Wei Wang, and Philip S. Yu. Mining Asynchronous Periodic Patterns in Time Series Data. *TKDE*, 15(3):613–628, 2003.

[127] Jiong Yang, Philip S Yu, and Unc-chapel Hill. InfoMiner+: Mining Partial Periodic Patterns with Gap Penalties. In *ICDM*, pages 725–728. IEEE Computer Society, 2002.

[128] Mao Ye, Dong Shou, Wang-Chien Lee, Peifeng Yin, and Krzysztof Janowicz. On the Semantic Annotation of Places in Location-based Social Networks. In *KDD*, pages 520–528. ACM, 2011.

[129] Jin Soung Yoo, Shashi Shekhar, John Smith, and Julius P. Kumquat. A Partial Join Approach for Mining Co-location Patterns. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 241–249. ACM, 2004.

[130] J.S. Yoo and S. Shekhar. A joinless approach for mining spatial colocation patterns. *TKDE*, 18(10):1323–1337, October 2006.

[131] Mohammed J. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1):31–60, 2001.

[132] Ying Zhao and George Karypis. Evaluation of Hierarchical Clustering Algorithms for Document Datasets. In *CIKM*, pages 515–524. ACM, 2002.