A Framework for Scalable Correlation of Spatio-temporal Event Data

Stefan Hagedorn[1](✉), Kai-Uwe Sattler[1], and Michael Gertz[2]

[1] Technische Universität Ilmenau, Ilmenau, Germany
{Stefan.Hagedorn,Kai-Uwe.Sattler}@tu-ilmenau.de
[2] Heidelberg University, Heidelberg, Germany
gertz@informatik.uni-heidelberg.de

Abstract. Spatio-temporal event data do not only arise from sensor readings, but also in information retrieval and text analysis. However, such events extracted from a text corpus may be imprecise in both dimensions. In this paper we focus on the task of event correlation, i.e., finding events that are similar in terms of space and time. We present a framework for Apache Spark that provides correlation operators that can be configured to deal with such imprecise event data.

1 Introduction

An event is often described as "something that happens at some place at some time". Thus, events inherently have a spatial and a temporal component. These spatio-temporal events do not only origin from sensor readings, but can also be extracted from text corpora like news, weblogs, and tweets.

The task we focus on in this paper is to find events that are correlated to a given event in terms of its time and place of occurrence. The result is, e.g., a list of pointers to documents in which similar events have been detected. For such correlation tasks, we are facing the following problems:

- First, event specifications are often imprecise. For example, for the event extracted from the sentence "Obama visited Germany in April 2009", we do not know (using only the text source) which part of Germany Obama visited or at what exact dates he visited Germany.
- Second, for comparing events in terms of their similarity solely based on their temporal and geographic components, we need a distance measure.
- Third, depending on the specific application different correlation techniques are needed: for finding similar events, nearest neighbor or skyline queries are an appropriate approach, whereas for determining hot spots, clustering (such as DBSCAN) might be a better choice.
- Finally, because (extracted) event data can be large datasets, scalable techniques are required. Modern data processing frameworks such as Apache Hadoop or Spark provide a suitable platform for addressing this challenge. In [2] an adaption of DBSCAN to MapReduce is proposed, whereas in [1] and [4] adaptions of the skyline algorithm are shown.

S. Maneth (Ed.): BICOD 2015, LNCS 9147, pp. 9–15, 2015.
DOI: 10.1007/978-3-319-20424-6_2

In this paper, we propose a framework that addresses the problem of determining the correlation between events. For this, we introduce an event model and indicate different distance measures for both the temporal and geographic components of events. We further introduce a set of basic operators for preparing as well as exploring and analyzing event data correlations. These operators are provided as transformation operators in Apache Spark and allow to define application-specific spatio-temporal event analysis pipelines including top-k and skyline processing as well as (density-based) clustering.

2 Event Data Model

We assume an event model in which information about events has been extracted from some document and is represented by a temporal and a geographic component along with other information like an ID and metadata such as the origin. The expressions underlying these components are based on concept hierarchies for time and space.

Temporal expressions can be of different granularities, with days being the finest and years the coarsest level of granularity. Although further granularities such as weeks or centuries can be included. For the sake of simplicity, in the following, we only focus on days, months, and years. We denote the corresponding domains as $T = \{T_{day}, T_{month}, T_{year}\}$.

Analogously, geographic expressions are taken from the domains in $G = \{G_{city}, G_{state}, G_{country}\}$. We assume that with each expression a spatial object in the form of a single polygon (without holes) is associated.

Definition 1. *(Event) Given concept hierarchies T and G for temporal and geographic expressions, respectively. An event $e = \langle t, g \rangle$ consists of a temporal expression t with $t.type \in T$ and a geographic expression g with $g.type \in G$.*

Examples of (imprecise) event specifications are (2013-09-02, Munich), (1955, Germany), or (2000-04, Bavaria). To account for these types of imprecision, in our framework we make the following assumptions:

1. Temporal and geographic expressions of the finest granularity are certain.
2. Every temporal (resp. geographic) expression of type P' that refines a given temporal (resp. geographic) expression of type P, with P' being of finer granularity than P, is equally likely.

Distance Measures. To compute correlations between events, we need a distance measure that takes both the temporal and the geographic component of an event into account, both of which can be imprecise. For the most fine-grained, point-based locations (e.g., cities) and days, this is trivial, resulting in a scalar value for time (e.g., distance in days) and location (e.g., distance in kilometers), which can be combined into some single (weighted) distance value. For events having an imprecise temporal or geographic expressions, different types of distance functions are meaningful and can be specified accordingly.

In general, there are two approaches for realizing a distance function for imprecise event data. First, dates representing a month or year can be mapped to intervals of days (e.g., "2014-05" can be mapped to [2014-05-01, 2014-05-30]) with each subinterval being valid instance of "2014-05". Similarly, a country can be mapped to a polygon or minimum bounding box. Then, a function is devised that determines the distance between intervals (for time) and boxes/polygons (for regions). Each such a function can either yield a single scalar value (e.g., the average distance between points of two intervals/boxes), or an interval, giving the minimum and maximum distance between two intervals/boxes. In our current framework, we only consider the former case where single scalar values for both the temporal and geographic component are determined and linearly combined using a weight. That is, for two events e_1 and e_2, we assume a distance function $dist(e_1, e_2) := w_t\ dist_t(e_1, e_2) + w_g\ dist_g(e_1, e_2)$, with $dist_e$ and $dist_g$ functions for determining the distance between intervals and regions/boxes, respectively, and $w_t, w_g \in [0, 1], w_t + w_g = 1$.

3 Techniques for Correlating Event Data

Correlating events means to find events in the dataset that have something in common or which have the same or a similar context. In this paper, we focus on the spatio-temporal aspect of events, which means we consider the similarity of events in terms of their spatial and/or temporal properties. Depending on the specific application different approaches can be used to determine correlations.

Nearest Neighbor Queries. Nearest neighbor queries represent the most straightforward solution. Given a set of events $\mathcal{E}$, a reference event e_r and a distance function *dist*, the task is to find the set $kNN(e_r)$ of the k nearest events. In the case of our spatio-temporal event data this requires a single distance measure, which is usually defined using weights for the spatial and temporal distances.

Skyline. Defining appropriate weights is often difficult. Skyline queries avoid this problem. Adapted to our scenario, the notion of the Skyline algorithm is to find those events in $\mathcal{E}$ that "match" a query event $q = \langle t_q, g_q \rangle$ best. Since we consider two dimension for events, time and space, it is thus intuitive to employ a skyline-based approach as there might be events that match t_q well but not g_q, and vice versa. A core concept of skylines is the dominance relationship. The skyline S_q consists of all events that are not dominated by any other event in $\mathcal{E}$ with respect to q. Because the dominance of an event with respect to another event is decided by their respective distances to q, the distance function outlined in the previous section come into play.

Clustering. Clustering represents another useful technique for correlating event data. Applied to the problem of event correlation we can form clusters of events on their distance values and, in this way, events belonging to the same cluster are considered to be correlated. Focusing only on the spatial and temporal dimension results in clusters of events that occur in close proximity in terms of space and time.

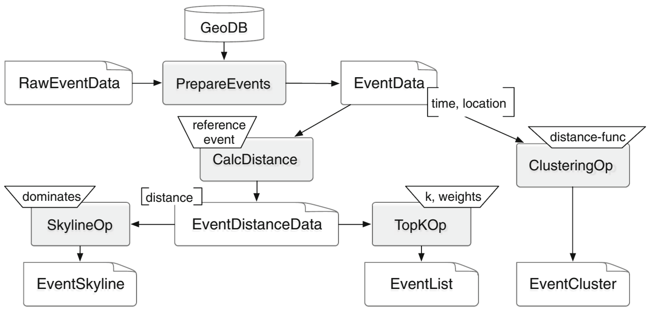


Fig. 1. Framework showing operators and event analysis pipeline

4 A Spark-Based Correlation Framework

Given the event data model, the distance functions, and the set of correlation functions described above, the goal of our work is to provide a framework for scalable event data correlation. As the underlying platform we have chosen Apache Spark[1], but our framework can be easily ported to other platforms providing a similar (Scala-based) API such as the Apache Flink[2] project. Figure 1 shows the components of the framework and their role in an event analysis pipeline.

The core components are the following operators implemented as transformations on Spark's resilient distributed datasets (RDD):

`PrepareEvents`: This operator transforms a set of raw (textual) event data into a set of event records $\langle t, q \rangle$ conforming to our framework. This means that textual temporal and spatial properties are normalized into numerical values, i.e., date/time values and points or polygons for the spatial descriptions such as names of cities or locations. For the latter, a service such as GeoNames[3] can be used.

`CalcDistance`: This implements a transformation operator for calculating the spatial and temporal distance *dist* of each event of a RDD to a given reference event.

`TopKOp`: This operator computes the top-k list of events from an input RDD produced by `CalcDistance`. Parameters to this operator are k as well as the weights for the geographic (w_g) and temporal (w_t) distance.

`SkylineOp`: This operator computes the skyline of event records from a RDD produced by `CalcDistance`. The dominance relation can be passed as parameter to the operator.

`ClusteringOp`: Finding groups of correlated events is realized by the `ClusteringOp` operator implementing a parallel variant of DBSCAN [3] for spatio-temporal data. Parameters are the standard clustering parameters ε

[1] http://spark.apache.org.
[2] http://flink.apache.org.
[3] http://www.geonames.org.

and MinPts as well as a global distance function taking both spatial and temporal distances into account.

While the implementation of `PrepareEvents`, `CalcDistance`, and – a sort-based – `TopKOp` operator is rather straightforward, efficient skyline processing and density-based clustering require more effort. As mentioned in Sect. 1, there already exist some proposals for MapReduce-based implementations of these operators that have inspired our Spark implementations.

Both `SkylineOp` and `ClusteringOp` are based on a grid partitioning, where the dimensions of the grid are either the spatial and temporal dimensions (in case of skyline processing) or longitude, latitude, and time in case of clustering. For simplicity, we assume – non-optimal – equally-sized grid cells representing partitions of Spark's RDDs.

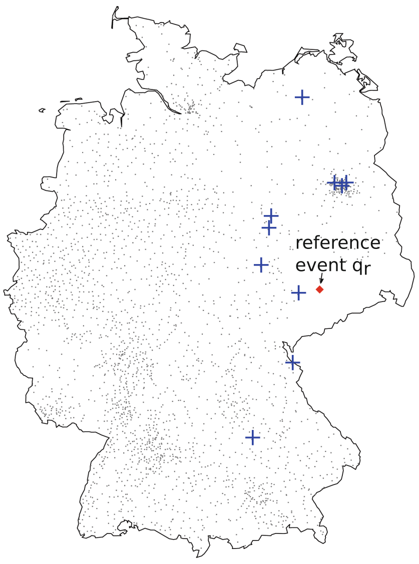
Our skyline operator implements the idea presented in [4] by computing in a first phase bitstrings representing grid cells containing data points. This can be done in parallel without moving data. By combining these bitstrings in a reduce step, dominated as well as empty cells can be pruned. In the second phase, all nodes compute a local skyline of their data by taking the information from this global bitstring into account. Finally, the local skylines are merged.

For density-based clustering, grid cells must not be disjoint in order to determine the neighborhood for objects at the border of cells. Thus, we compute an overlap between neighboring cells and assign objects in this overlap area to its neighbor cells, too. Next, for each cell a local DBSCAN is performed. Note that compared to the skyline processing strategy, this requires to repartition data according their grid cell membership. Finally, we build a global graph of all local clusters in order to merge clusters from different cells.

5 Use Cases

In this section, we show the outcome of the skyline and top-k operations. Due to space limitations we do not present a full performance evaluation. Our test dataset was crawled from the website `eventful.com` and contains 122,467 events. It consists only of events that took place in Germany where the earliest event appeared on 2007-06-30 and the latest on 2020-06-30. For the test of our operators, we manually removed all events in the eastern part of Germany (which is the federal state of Saxony).

Figure 2 shows the spatial distribution of all events in our dataset. On the left, the skyline (marked with +) is shown. The right figure shows the result of the top-k query ($k = 10$; marked with $\bullet$). The reference point for both queries is shown as ⧫. One can see that the spatio-temporal skyline not only finds correlated events that have both a small spatial and temporal distance to the reference event, but also considers events as correlated that are near to the reference event in at least one dimension. The two shown skyline points in the north and the south have a large spatial distance, but only a small temporal distance and thus, are considered correlated to the reference event. On the other hand, the top-k operator accepts user-defined weights for the spatial and temporal distances



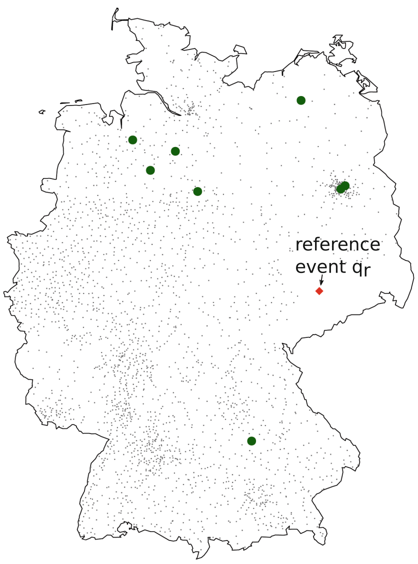


Fig. 2. Left: the skyline (+); right: top-10 result (●) for a reference event (⧫).

to express a desired preference over one or the other dimension. In the given example these weights are $w_g = 0.10$ for the geographic and $w_t = 0.90$ for the temporal dimension, i.e., the temporal distance is considered more important. As Fig. 2 shows, the resulting points have a large geographic distance, but are near to the reference event in the temporal dimension. Note, there are events that take place at the exact same position, so that they cover each other in the figure and appear as one point. Thus, the figure shows only eight result points. Due to space limitations, we cannot show the results of the spatio-temporal clustering.

6 Conclusions and Ongoing Work

In this paper, we presented a framework for Apache Spark that provides operators for computing correlated events. We provide operators for data import and cleaning as well as operators for the actual correlation tasks. These operators can be configured by their parameters and the distance function - for which we also provide several alternatives. Our ongoing work focuses more on imprecise data and respective distance functions that return intervals instead of scalar values, which will result in, e.g., SkyBands instead of Skylines.

Acknowledgement. This work was funded by the DFG under grant no. SA782/22.

References

1. Chen, L., Hwang, K., Wu, J.: MapReduce skyline query processing with a new angular partitioning approach. In: IPDPSW (2012)
2. Dai, B.-R., Lin, I.-C.: Efficient map/reduce-based DBSCAN algorithm with optimized data partition. In: CLOUD (2012)
3. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD (1996)
4. Mullesgaard, K., Pederseny, J.L., Lu, H., Zhou, Y.: Efficient skyline computation in MapReduce. In: EDBT (2014)