Large-scale Analysis of Event Data

Stefan Hagedorn
TU Ilmenau, Germany
stefan.hagedorn@tu-ilmenau.de

Kai-Uwe Sattler
TU Ilmenau, Germany
kus@tu-ilmenau.de

Michael Gertz
Heidelberg University, Germany
gertz@informatik.uni-heidelberg.de

ABSTRACT

With the availability of numerous sources and the development of sophisticated text analysis and information retrieval techniques, more and more spatio-temporal data are extracted from texts such as news documents or social network data. Temporal and geographic information obtained this way often form some kind of event, describing when and where something happened. An important task in the context of business intelligence and document exploration applications is the correlation of events in terms of their temporal, geographic or even semantic properties. In this paper we discuss the tasks related to event data analysis, ranging from the extraction of events to determining events that are similar in terms of space and time by using skyline processing and clustering. We present a framework implemented in Apache Spark that provides operators supporting these tasks and thus allows to build analysis pipelines.

1. INTRODUCTION

Traditionally, research on querying and analyzing spatio-temporal data focuses on data obtained from sensors that record the evolution and movement of objects in 2- or 3-dimensional space over time. Typical examples of such data include trajectories of moving objects (e.g., [7]) and remotely-sensed data (e.g., [10]). Spatio-temporal data, however, do not only arise in the context of sensor data or, more generally, from observing objects with their spatial extent over time. In this paper, we consider *events* as an important type of spatio-temporal data that thus far have been neglected in the context of data analysis. Events play an important role in information retrieval and text analysis in general, most prominently in the task of topic detection and tracking (TDT) [2, 13]. An event is often described as “something that happens at some place at some time”, e.g., [24]. Thus, events inherently have a spatial and a temporal component.

A prominent source for event data are textual information from newsfeeds, websites, and social media such as blogs, tweets or social networks. Underlying the extraction of such information about events are temporal taggers for the temporal component and geo-taggers for the spatial or geographic components [21]. Such taggers detect, extract, and normalize respective expressions in textual data and provide subsequent tools as the basis for further tasks such as document clustering or classification. For example, from the sentence “Obama visited Germany in April 2009”, a temporal tagger would detect the expression “April 2009” and normalize it to “2009-04”; similarly, a geo-tagger would extract the expression “Germany” and often associate further information with it. This might include the spatial extent in the form of a polygonal description or that Germany is part of Europe (using some concept hierarchy). Such a pair of temporal component and geographic component then forms an event. Given that today’s taggers become more and more sophisticated, large repositories of event information can be built from news articles, blog entries, social network postings, and medical records, to name but a few. 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 of such a query is a list of pointers to documents in which similar (correlated) events have been detected.

For such correlation tasks, one is facing several challenges ranging from extracting event data from documents, dealing with imprecise event specifications resulting from the extraction process, to exploiting different correlation techniques for finding similar events, to scalable processing for large datasets.

In this paper, we describe a framework for analyzing event data and detecting correlations between events. Based on a model for spatio-temporal events at different granularities we discuss corresponding distance measures. We present the analysis tasks and discuss how these tasks can be supported by a set of basic operators for extracting event data from text documents, preparing the data, and analyzing event correlations using top-k and skyline processing as well as clustering. We further describe how these operators are supported as transformation operators in Apache Spark and outline the support for explorative analyses.

2. EVENT MODEL

An important data analysis task is to find similar events, i.e. events that share the same context or have other values in common. In this paper, we consider the spatio-temporal aspect of events for determining similarities, i.e., we focus on the similarity of their respective location and/or time of occurrence.

27th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 26.05.2015 - 29.05.2015, Magdeburg, Germany.

For our analysis framework, we assume an event model in which information about events has been extracted from some document (see Sect. 4) and is represented by useful and necessary information like ID, description, origin, etc. as well as a temporal and a geographic component, describing the when and where. The expressions underlying these components are based on concept hierarchies for time and space, as shown in Figure 1.

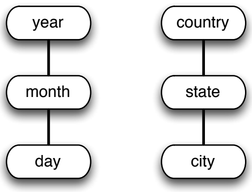


Figure 1: Concept hierarchies for temporal and geographic information

The temporal as well as the spatial expressions can be of different granularities. For temporal expressions we consider days to be of the finest and years of the coarsest granularity. Of course one could also extend this model with further granularity levels, such as weeks, hours, or minutes. In this paper, however, and for the sake of simplicity, we will only consider days, months, and years. We denote the corresponding domains as $T = \{T_{day}, T_{month}, T_{year}\}$. For example, "2001-05-20", "2013-05", and "1995" are all valid temporal expressions from these three different domains.

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. For example, the geographic expressions "Heidelberg" and "Germany" are both valid expressions of type G_{city} and $G_{country}$, respectively.

Definition. (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 uncertainties, 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 determine the similarity between events, a distance measure is needed that takes both the temporal and the geographic component of an event into account, both of which can be imprecise.

Precise events are those events for which both the location and temporal information is given as a fine-grained concept, i.e., they are defined on the city and day level. For such events, calculating the distance is trivial resulting in a scalar value for time (e.g., distance in days) and location (e.g. distance in meters). Both values can be combined into a single (weighted) result. Although we cannot use the traditional distance functions for events that are imprecise in at least one component, we can specify new distance functions accordingly. However, the definition of such functions is beyond the scope of this paper and we will give only a brief overview below.

First, we convert the imprecise event data into intervals and regions. As imprecise temporal data is defined as a month or year (to be imprecise the day part is missing), we can create an interval of days that starts with the minimal possible day, i.e., the first day of the corresponding month or a year and ends with the maximal possible day, which is the last day of the month or year, respectively. Each subinterval is called a valid instance of this interval. As an example consider the imprecise temporal expression "2014-05" that is mapped to the (right-open) interval i = [2014-05-01, 2014-05-30]). Note that any subinterval in i is a valid instance of the temporal expression. Analogously, for imprecise geographic expression, i.e., expressions not at the city level, a polygon (or its minimum bounding box) is used. Then, a function that calculates the distance between such intervals and between polygons/boxes is needed. Such distance function can yield a scalar value, which may be the average distance between points in the respective intervals or polygons, or it can yield an interval, representing the minimum and maximum distance between the intervals/polygons.

The Hausdorff distance is a typical approach to calculate the distance between two intervals or polygons as a scalar value. Its main idea is to compute the maximum distance between any point in one interval to any other point of the second interval. For two temporal intervals t_1 and t_2, the distance can be computed as

$$d_H(t_1, t_2) := \max\{\max_{i_1 \in t_1} d_h(i_1, t_2), \max_{i_2 \in t_2} d_h(i_2, t_1)\}$$

where the distance $d_h(i, t)$ between a day i and a temporal interval t defined as $d_h(i, t) := \min_{i' \in t}(|i - i'|)$.

For polygons/boxes, the Hausdorff distance can be defined as follows:

$$d_H(g_1, g_2) := \max_{x \in g_1} \min_{y \in g_2} ||x - y||$$

This is the greatest distance from a point in g_1 to the closest point in g_2. However, the calculation of this distance measure can become very expensive as the distance from every point in the first interval to every other point in the second interval has to be computed. For temporal intervals this is not a big problem, because when taking years as the most imprecise and days as most precise level, such an interval can have only 365 (or 366) points at most. However, for polygons the set of inner points is in principle infinite. On the one hand, one could use the cities that are located in the respective region. This approach may not lead to good results as many cities may be clustered around a large metropolis, whereas in rural regions only very few cities can be found. On the other hand, one could impose a raster on the respective regions using a fixed grid. Then, only the points of this grid will be used for calculations. Then, however, the definition of the grid size may be difficult since choosing a small grid step size may lead to many data points that will take part in the calculation and thus, will not improve the computation time. Choosing a large grid size will result in only very few data points, which might again result in incorrect distance values. In order to use common distance functions, one could reduce a polygon to a single point that represents this polygon, e.g., the center point. However, these simplifications result in distances that may not necessarily reflect the real distances. Therefore, more sophisticated distance functions are needed.

3. ANALYSIS TASKS

Analysis of event data comprises several tasks. Depending on the sources of events (structured data, text documents) and the specific questions different tasks have to be combined in an analysis pipeline. In addition, such analyses are often interactive and explorative processes requiring a flexible combination of a set of powerful extraction and correlation operators. The whole process is depicted in Fig. 2. In the following, we describe a basic set of tasks that our framework supports by dedicated operators.

3.1 Event Information Extraction

Prior to any analysis task event information needs to be extracted from the text documents of a given corpus, such as Wikipedia or news articles. As an event is assumed to be composed of a temporal expression describing the "when" of an event and a geographic expression describing the "where" of an event, respective event components need to be determined in a given text and mapped to some standard format. The function of a temporal tagger is to determine such temporal information and to map each piece of information found to a temporal expression. One typically distinguishes between explicit, implicit and relative information temporal information. Explicit temporal information refers to a sequence of tokens that describe an exact date, month in a year or year, e.g., "April 1, 2015". Implicit temporal information often refers to holidays or some named events, such as "Independence Day 2015" or "Summer Olympics 2012". Finally, relative temporal information can only be determined using the context or document in which the token sequence appears. For example, in this paper, the string "last year" would refer to the year 2014. Popular temporal taggers such as HeidelTime [21] are able to detect such information and map them to a standard format, which is mostly based on the TIMEX3 annotation standard. For example, the above explicit temporal information "April 1, 2015" would be mapped to the temporal expression "2015-04-01".

Similarly, for geographic information in documents, a geo-tagger is employed. Popular tools include GeoNames[1] or Yahoo! Placemaker[2]. Mapping token sequences found in a text to some standardized format, however, is less trivial. For example, most taggers would map the string "Magdeburg" to a latitude/longitude pair rather than to some complex polygon description. Also, several geo-taggers also include some information about the granularity of the geographic expression based on concept hierarchies.

Once a temporal tagger and a geo-tagger have extracted and mapped temporal expressions from a given text, events are formed. In the most simple case, an event is a pair consisting of a temporal expression and a geographic expression found in close text proximity, typically at the sentence level.

All events extracted from a document or document collection are then stored based on our event model described in Sect. 2. For each event detected, it describes the normalized temporal and geographic expression, the granularity of the expressions, and also some offset information (in what document and at what position each of the two components have been determined).

3.2 Correlation Analysis

Correlations on event data can be computed in different ways, depending on specific applications. In the following, we discuss the three operations for correlation computations.

Nearest Neighbor Queries.

A straightforward solution to find correlated, i.e., similar, events is to find the neighbors of a given reference event. 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. These weights are necessary, because we cannot directly combine the spatial distance with the temporal distance into one distance measure. However, with the weights we can adjust the impact of the spatial and temporal distance to the overall distance:

$$dist(e_1, e_2) = w_g \cdot dist_g(e_1, e_2) + w_t \cdot dist_t(e_1, e_2)$$

where $w_t, w_g \in [0, 1]$ and $w_t + w_g = 1$.

Skyline.

In contrast to the Nearest Neighbor search, Skyline queries do not need weights for the spatial and temporal distance, which are often difficult to determine. 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:

$$S_q = \{e_i | e_i \in \mathcal{E} \wedge \neg\exists e_j \in \mathcal{E} : e_j \neq e_i \wedge e_j \succ_q e_i\}$$

where $\succ_q$ denotes a dominance relationship with $e_j \succ_q e_i$ meaning that e_j is "in closer proximity to q" than e_i. Because the dominance of an event with respect to another event is determined based on their respective distances to q, the distance function outlined before comes into play.

Clustering.

Other than in the two approaches mentioned before, for clustering, a definition of a reference event is not needed. It is typically understood as the task of grouping objects based on the principle of maximizing the intra-class similarity and minimizing the inter-class similarity. However, there is a rich diversity in specific definitions of clustering and many different techniques exist [1].

The clusters can be formed using distance values between other events. Thus, events that belong to the same cluster can be considered to be correlated as they occur around the same time and place.

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

[2]`https://developer.yahoo.com/boss/geo/`

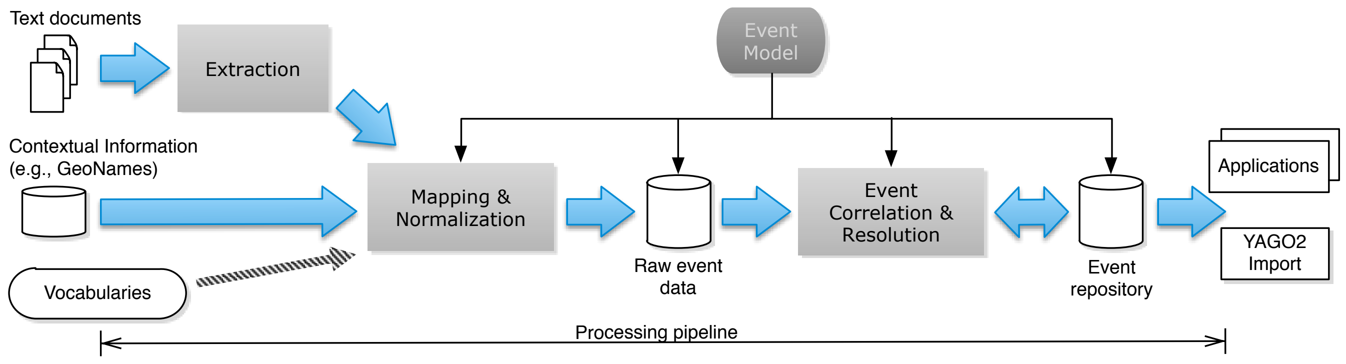


Figure 2: Overview of event analysis pipeline.

4. PROCESSING PIPELINE

After the events have been extracted from the text document using the approach as outline in Section 2, the events are fed into the correlation pipeline. This pipeline is implemented using the Apache Spark[3] platform. However, it can be easily ported to other platforms like Apache Flink[4] or even Google's new DataFlow SDK[5] if they provide a Scala-based API. We chose Spark over other MapReduce based approaches, e.g., Pig, because it provides a very efficient data structure called resilient distributed datasets (RDD). RDDs are immutable, in-memory partitioned collections that can be distributed among the cluster nodes. With the help of such a data structure, the performance of the computations is significantly faster than in MapReduce programs that materialize their intermediate results to disk. Furthermore, Spark allows to implement iterative models that are needed for our computations as well as for programs following the MapReduce paradigm.

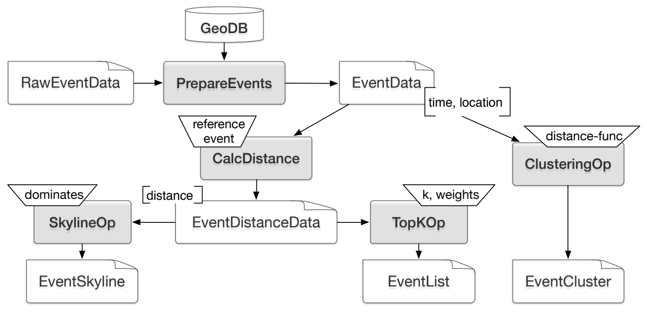


Figure 3: Framework overview

The Spark operators represent transformations on these RDD. Fig. 3 gives an overview of the correlation pipeline, where the tasks of the operators is described below:

`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 geographic 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.

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

`TopKOp`: This operator computes the k nearest neighbors as a 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 an RDD produced by `CalcDistance`. Our implementation adopts the grid-based approach using bitsets described in [14] for the Spark platform. 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 [9] for spatio-temporal data. Parameters are the standard clustering parameters ε and MinPts as well as a global distance function taking both geographic and temporal distances into account.

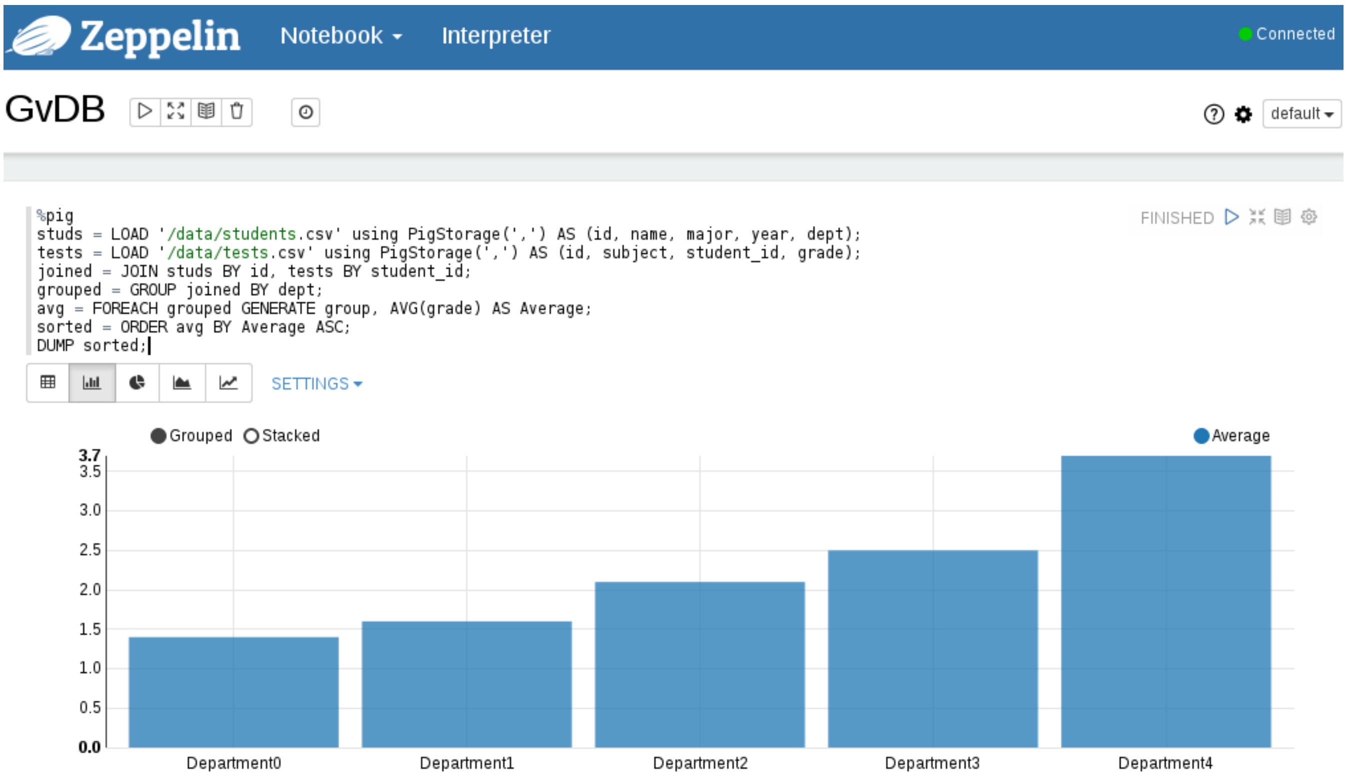
5. TOWARDS AN INTERACTIVE EXPLORATION OF EVENT CORRELATIONS

We will use the above processing pipeline to build an event repository that makes the extracted event data publicly available as Open Data. As an underlying data structure we are going to use the Linked Data principle, which allows for a flexible schema for different types of events and also makes the information contained in the event itself machine readable. Furthermore, one can easily add links between events that have been identified to be correlated and thus make these correlations also available for querying and processing. Adding links to other datasets like YAGO2 or DBpedia will enrich our event repository with even more useful and broader information that can be used for data analysis and exploration.

The event repository will be free to download and will also be made queryable via web services. Via an API users can run Spark programs (using operators introduced in Sect. 3) or SPARQL queries.

Next to the event repository we plan to develop a data exploration platform that will allow users to interactively analyze the data. Different from traditional business intelligence frameworks that provide only a predefined set of tools

[3]`http://spark.apache.org`
[4]`http://flink.apache.org`
[5]`https://cloud.google.com/dataflow/`



```
%pig
studs = LOAD '/data/students.csv' using PigStorage(',') AS (id, name, major, year, dept);
tests = LOAD '/data/tests.csv' using PigStorage(',') AS (id, subject, student_id, grade);
joined = JOIN studs BY id, tests BY student_id;
grouped = GROUP joined BY dept;
avg = FOREACH grouped GENERATE group, AVG(grade) AS Average;
sorted = ORDER avg BY Average ASC;
DUMP sorted;
```

Figure 4: Screenshot of Zeppelin after executing a Pig script. The results are visualized as bar chart.

for reporting and visualization, our planned platform allows very different interaction paradigms. We integrate the idea of notebooks that was inspired by Mathematica's interactive documents and IPython's (and Jupyter's) notebooks. With the help of such notebooks users can create their own programs that fit their needs and run them in a managed cluster environment without the need to set up any hardware or other software before. The idea of web-based notebooks allows users to organize text, scripts, and plots on a single webpage so that all information for data analytics can be kept in one place.

For our data exploration tool we chose the Apache Zeppelin[6], because it already has support for Spark programs and also provides the possibility to visualize script results. Content in the format of CSV, HTML, or images can directly be visualized as tables, bar charts, line graphs, and scatter plots. A notebook in Zeppelin is organized in paragraphs where each of them can contain and execute a script of a different (supported) language. On execution, the script content of a paragraph is sent to the server, which will forward it to the appropriate interpreter. The interpreter is chosen by a magic keyword given by the user at the beginning of the script, e.g. `%spark`. When the execution has finished, the interpreter will return the results to the server, which in turn will publish them on the website. Figure 4 shows a screenshot of a notebook in Zeppelin that executed a Pig [16] script and shows the results of that script in a bar chart.

Zeppelin currently supports shell scripts and Spark programs written in Scala or Python out of the box and can also make use of Spark modules like SparkSQL and Spark Streaming. Although the Spark support lets users create efficient data analytics programs, it may be hard for a non-programmer to create such programs.

We argue that a declarative approach will be easier to use for users that are not familiar with the Spark API and the concepts of their RDDs. For this reason, we integrate a new interpreter that allows to run Pig scripts. Since our event repository uses Linked Data, we do not use the conventional Pig interpreter, but our extended Pig version that allows to use SPARQL-like features, such as BGP, in Pig scripts [11]. To do so, we enhanced Pig's `FILTER` operator. This allows the user to easily define queries on the Linked Data sets with a more powerful language than SPARQL. Though, the triple structure of the Linked Data concept is not very well suited for a tuple-based execution environment as it requires a lot of (self-)joins to reconstruct all details of an entity from the triples. To solve this issue, in [11] we introduced a new triple bag format (based on the concept introduced in [12]) that keeps the flexibility of triples with the convenience of tuples. We also showed that this format can lead to significant performance gains.

However, since Pig programs are compiled into MapReduce programs, the execution of these programs will take longer than for Spark programs. Thus, to combine the advantages of both approaches, in our ongoing work we use the Pig Latin language and compile it into Spark programs. This will allow users to write intuitive data flow scripts without the need to know a specific API and characteristics of the underlying platform and to get very efficient programs that will execute faster than MapReduce programs.

[6]`https://zeppelin.incubator.apache.org`

6. RELATED WORK

For our event correlation framework, we employ concepts developed in different areas of information extraction, event similarity especially for spatio-temporal similarity, as well as skylines and clustering algorithms for distributed environments.

To extract the spatial and temporal information from textual data, we use concepts that were described, e.g., in [22, 23]. In addition to these works, the notions of event similarity and relationships between events has also been studied in the context of social media, e.g., [3, 17, 18].

Our correlation operators mainly focus on Skylines and clustering. Among the numerous techniques that build on the concept of skyline queries [5], there also has been some work that study skylines for geographic and uncertain or probabilistic data. These include spatial skyline queries [19, 20] where no temporal aspects are considered for data points.

To compute skylines for large datasets, new algorithms have been proposed that allow the computation in MapReduce environments. Here, an important challenge is to partition the data in a way, that the partitioning itself is efficient and data is distributed to all nodes equally to ensure that all nodes get approx. the same workload to avoid one node having to do all the work while the others are idle. In [14] an approach using a grid based partitioning is introduced, and [6] shows an angular based partitioning approach.

For clustering, density based algorithms like DBSCAN [9] are chosen if the number of clusters is not known apriori. For dealing with spatio-temporal data an extension to DBSCAN has been proposed in [4], that uses two ε parameters (one for spatial and one for temporal distance). To run the clustering algorithms in a distributed environment for MapReduce, we rely on concepts developed in [15, 8].

7. SUMMARY

In this paper we presented our approach for an event correlation framework based on Apache Spark. The framework provides operators for importing data as well as for clustering, skylines and k nearest neighbor queries. We chose Apache Zeppelin for our exploration tool to allow an incremental creation of scripts and an immediate visualization of results.

Acknowledgements

This work was funded by the German Research Foundation (DFG) under grant no. SA782/22.

8. REFERENCES

[1] C. C. Aggarwal and C. K. Reddy. *Data clustering: algorithms and applications*. CRC Press, 2013.

[2] J. Allan, editor. *Topic detection and tracking: event-based information organization*. Kluwer Academic Publishers, 2002.

[3] H. Becker, M. Naaman, and L. Gravano. Learning similarity metrics for event identification in social media. In *WSDM*, pages 291–300, 2010.

[4] D. Birant and A. Kut. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 2007.

[5] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[6] L. Chen, K. Hwang, and J. Wu. MapReduce Skyline Query Processing with a New Angular Partitioning Approach. In *IPDPSW*, May 2012.

[7] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, 2005.

[8] B.-R. Dai and I.-C. Lin. Efficient Map/Reduce-Based DBSCAN Algorithm with Optimized Data Partition. In *CLOUD*, June 2012.

[9] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *KDD*, 1996.

[10] A. Ganguly, O. Omitaomu, Y. Fang, S. Khan, and B. Bhaduri. Knowledge discovery from sensor data for scientific applications. In *Learning from Data Streams*. 2007.

[11] S. Hagedorn, K. Hose, and K.-U. Sattler. Sparqling pig - processing linked data with pig latin. In *BTW*, March 2015.

[12] H. Kim, P. Ravindra, and K. Anyanwu. From SPARQL to MapReduce: The Journey Using a Nested TripleGroup Algebra. *PVLDB*, 4(12):1426–1429, 2011.

[13] J. Makkonen, H. Ahonen-Myka, and M. Salmenkivi. Topic detection and tracking with spatio-temporal evidence. In *Advances in Information Retrieval*, volume 2633, pages 251–265. 2003.

[14] K. Mullesgaard, J. L. Pederseny, H. Lu, and Y. Zhou. Efficient Skyline Computation in MapReduce. In *EDBT*, 2014.

[15] M. Noticewala and D. Vaghela. MR-IDBSCAN: Efficient Parallel Incremental DBSCAN Algorithm using MapReduce. *Intl J Computer Applications*, 93(4):13–17, 2014.

[16] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: a not-so-foreign language for data processing. In *SIGMOD*, 2008.

[17] W. Ribarsky, D. Skau, X. Wang, W. Dou, and M. X. Zhou. Leadline: Interactive visual analysis of text data through event identification and exploration. *VAST*, 0:93–102, 2012.

[18] A. D. Sarma, A. Jain, and C. Yu. Dynamic relationship and event discovery. In *WSDM*, pages 207–216, 2011.

[19] M. Sharifzadeh, C. Shahabi, and L. Kazemi. Processing spatial skyline queries in both vector spaces and spatial network databases. *TODS*, 2009.

[20] W. Son, M.-W. Lee, H.-K. Ahn, and S. won Hwang. Spatial skyline queries: An efficient geometric algorithm. In *SSTD*, pages 247–264, 2009.

[21] J. Strötgen and M. Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013.

[22] J. Strötgen and M. Gertz. Proximity2-aware ranking for textual, temporal, and geographic queries. In *CIKM*, pages 739–744, 2013.

[23] J. Strötgen, M. Gertz, and C. Junghans. An event-centric model for multilingual document similarity. In *SIGIR*, pages 953–962, 2011.

[24] Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *SIGIR*, pages 28–36, 1998.