Real-Time Discovery and Geospatial Visualization of Mobility and Industry Events from Large-Scale, Heterogeneous Data Streams

Leonhard Hennig[*], **Philippe Thomas**[*], **Renlong Ai**[*], **Johannes Kirschnick**[*], **He Wang**[*], **Jakob Pannier**[†], **Nora Zimmermann**[†], **Sven Schmeier**[*], **Feiyu Xu**[*], **Jan Ostwald**[†], **Hans Uszkoreit**[*]

[*]DFKI, Berlin, `firstname.lastname@dfki.de`

[†]DB Systel, Berlin, `firstname.lastname@deutschebahn.com`

Abstract

Monitoring mobility- and industry-relevant events is important in areas such as personal travel planning and supply chain management, but extracting events pertaining to specific companies, transit routes and locations from heterogeneous, high-volume text streams remains a significant challenge. We present *Spree*, a scalable system for real-time, automatic event extraction from social media, news and domain-specific RSS feeds. Our system is tailored to a range of mobility- and industry-related events, and processes German texts within a distributed linguistic analysis pipeline implemented in Apache Flink. The pipeline detects and disambiguates highly ambiguous domain-relevant entities, such as street names, and extracts various events with their geo-locations. Event streams are visualized on a dynamic, interactive map for monitoring and analysis.

1 Introduction

Monitoring relevant news and events is of central importance in many economic and personal decision processes, such as supplier/supply chain management, market research, and personal travel planning. Social media, news sites, but also more specialized information systems such as on-line traffic and public transport information sources, provide valuable streams of text messages that can be used to improve decision making processes. For example, a sourcing department of a company may wish to monitor world-wide news for disruptive or risk-related events pertaining to their suppliers (e.g. natural disasters, strikes, liquidity risks), while a traveler wants to be informed about traffic events related to her itinerary (e.g. delays, cancellations). We thus want to extract and recognize events from message streams that mention very specific entities, such as companies, locations, or routes. For example, from the sentence "On Friday, Amazon employees once more went on strike in the company's shipping center in Leipzig.", we would like to extract a *strike* event with arguments *time=Friday*, *organization=Amazon*, and *location=Leipzig*.

Detecting such events in textual message streams raises a number of challenges. Social media streams, such as Twitter, are of very high volume and often contain duplicated, imprecise and potentially non-trustworthy information. They are written in a very informal, not always grammatically well-formed style (Osborne et al., 2014), which cannot easily be processed with standard linguistic tools. News sites provide well-formed texts, but their content is very heterogeneous and often hard to separate from non-relevant web page elements. Both types of sources relate information about an unbounded number of topics, which means that relevant messages need to be distinguished from irrelevant data. Domain-specific information sources, on the other hand, are topic-focused, but employ a wide variety of formats, from telegraph style texts to table entries. Processing such sources hence often requires customized analysis pipelines as well as domain adaptation of existing linguistic tools. We also typically require that documents are processed in (near) real-time in order to enable timely responses to important events. Finally, utilizing domain-specific, large entity datasets raises additional challenges for named entity recognition and linking, including significantly more cross-type ambiguities (e.g. synonymous street and transport stop names) and a higher rate of ambiguous, long-tail entities (e.g., there is a "main street" or "bus route #1" in many towns).

We introduce *Spree*, a scalable platform for real-time, fine-grained event extraction and geospatial visualization (Section 2). It is designed to process German texts from social media, news, web sites and traffic information sources in a distributed, scalable, and fault-tolerant manner (Section 3). These features are realized by implementing the system's main components within the Apache Flink framework (Alexandrov et al., 2014), a big data analytics platform which provides a high through-put, streaming computing environment. Events are automatically detected and geo-located within a linguistic analysis pipeline (Section 4), and visualized for the end user in a web-based application (Section 5). The system in its current version is tailored to the mobility and supply chain domains, and allows users to monitor and analyze messages for a range of events.

2 System Overview and Architecture

The system was designed based on the following requirements:

- Combine mobility and industry data from structured knowledge resources with information from highly dynamic, unstructured or semi-structured document streams
- Identify and extract mobility and industry-related entities and events from dynamic data streams
- Enable large-scale, fault-tolerant processing of data streams in (near) real-time
- Visualize identified events on a dynamic, interactive map

Figure 1 gives an overview of the system architecture. The system separates data ingestion (crawling) from the actual processing, which facilitates the integration of new data sources. Data ingestion and processing are loosely coupled via a message queue (MQ). The streaming processing of documents encompasses three major tasks: Preprocessing of documents (including HTML-to-text conversion, boilerplating, and linguistic preprocessing), entity discovery and linking, and event extraction. Components are coupled with a shared document schema which stores annotation results. Annotated documents are persisted to a distributed index. Additional structured knowledge is stored in relational databases, and served via REST interfaces. The user interface is implemented as a light-weight web application.

3 Data Sources

This section describes the data streams and knowledge resources that are implemented in the *Spree* system. Table 1 shows summary statistics of the data streams, and Table 2 shows summary statistics of the knowledge resources.

Data	Size
Tweets	11.5 M
News	1.2 M
RSS items	12.4 M

Table 1: Data statistics, Jan 1st – Mar 31st, 2016

Type	# Concepts	Resource
Company	112,347	Internal Dataset
City	27,075	OpenStreetMap
Street	104,598	OpenStreetMap
Station	9,860	Deutsche Bahn
Route	25,907	Deutsche Bahn

Table 2: Data statistics for knowledge resources

3.1 Data Streams

All source data streams are handled with individual crawlers, which push documents to Kafka MQ[1] for further processing by the analysis pipeline (see Section 4).

Twitter We use the Twitter Search API[2] to obtain a topically focused streaming sample of tweets. For our current system, we define the search filter using a list of approximately 150 domain-relevant users/channels and 300 search terms. Channels include e.g. airline companies, traffic information sources, and railway companies. Search terms comprise event-related keywords such as "traffic jam" or "roadworks", but also major highway names, railway route identifiers, and airport codes. This limits the number of tweets to approximately 50.000 per day, at a rate of about 35 tweets a minute.

News We retrieve news pages and general web

[1]`kafka.apache.org`
[2]`dev.twitter.com/rest/public/search`

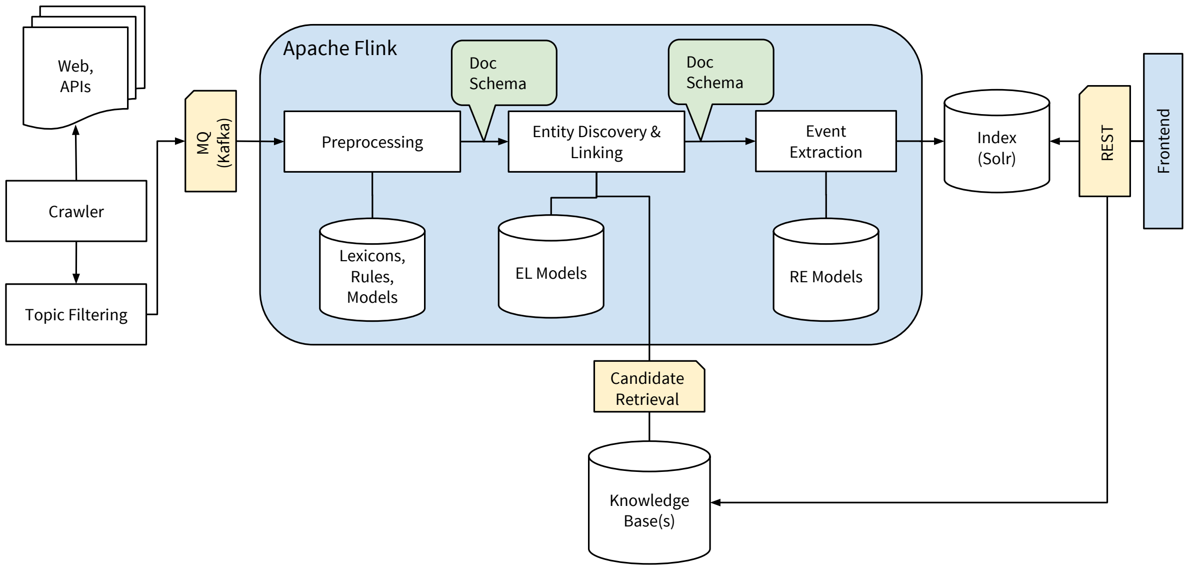


Figure 1: System architecture

sites using the uberMetrics Search API[3], which provides an interface to more than 400 million web sources that are crawled on a regular basis. The API allows us to define complex boolean search queries to filter the set of web pages that our system needs to process. We employ the same search terms as for Twitter, and limit the language to German. This provides approximately 13.200 documents per day (9/min).

RSS Feeds We implemented custom crawlers for a representative set of approximately 100 RSS feeds that provide traffic and transportation information. Feed sources include federal and state police, radio stations, and air travel sources. The feeds are fetched at regular intervals, yielding approximately 136.000 feed items per day (95/min).

3.2 Knowledge Bases

We integrate several types of knowledge resources for domain-specific named entity recognition, entity linking and event extraction. All resources are stored in a PostgreSQL relational database.

Companies We maintain a list of approx. 800K German companies, which includes many small and medium enterprises. From this list we selected a subset of 112,347 companies with ≥ 20 employees. Company entries comprise information about the company's name, judicial form, number of employees, and industry sector, and are associated with one or more postal addresses which were geocoded using Nominatim[4]. If applicable, entries are linked to DBpedia[5] and Freebase[6].

OpenStreetMap OpenStreetMap provides data dumps[7] that we use as a knowledge resource for location names. In particular, we utilize city, town, village, highway and road data for Germany, including names and geo-shape information.

GTFS Our final resource consists of transportation data of the German railway company Deutsche Bahn AG. This dataset contains railway and public transport stops, timetables, and transit routes, and includes geographical information for each entity. The data is provided in the commonly used General Transit Feed Specification (GTFS) format[8].

4 Document Processing Pipeline

This section describes the implementation of the streaming pipeline to process documents in the *Spree* system.

4.1 Data Schema

We store analysis results of individual pipeline components in a shared document schema that is inspired by the Common Analysis Structure (CAS) approach implemented in UIMA (Ferrucci and Lally, 2004). The schema defines elements,

[3]`doc.ubermetrics-technologies.com/api-reference/`

[4]`nominatim.openstreetmap.org`
[5]`dbpedia.org`
[6]`freebase.com`
[7]`planet.openstreetmap.org`
[8]`developers.google.com/transit/gtfs`

such as documents, sentences, tokens, concepts and relations. Annotations are either realized as attributes of these classes, or in a generic fashion using a labeled span scheme. We use Avro[9], a compact binary data format, to serialize documents between pipeline processing steps.

4.2 Stream Processing

Document processing is implemented in an analysis pipeline that is realized within the Apache Flink framework (Alexandrov et al., 2014)[10]. Flink's core is a streaming data flow engine that provides data distribution, communication, and fault tolerance for distributed computations over data streams. All document processors ingest messages (documents) from the MQ system. This architecture ensures data durability via Kafka's replication and disk persistence features, and consistent data movement and computation with Apache Flink. Together, these frameworks guarantee exactly-once delivery of events, can handle back pressure in case of data stream peaks, and provide high throughput. Our system thus can easily scale to larger and faster data streams than those we currently handle, for example when extending the system to monitor more data streams, other languages, or a wider range of events.

Our document processing implementation reads messages from Kafka and converts them to the internal document schema. For news and web sites, we extract text from the HTML and remove boilerplate code[11]. We perform language detection, and discard any non-German documents. All remaining documents are then passed to the linguistic analysis components, described next.

4.3 Linguistic Analysis

The linguistic analysis components process documents to detect and geo-locate mobility- and industry-related entities and events.

Documents are first segmented into sentences and tokens. We use the Mate Tools suite (Bohnet, 2010) to part-of-speech tag words, and for dependency parsing of sentences. For named entity recognition, we utilize SProUT (Drozdzynski et al., 2004), which implements a regular expression-like formalism and gazetteers for detecting concepts in text. SProUT uses rule sets to deal with frequent morphologic variations and abbreviations, e.g. "strasse" and "straße" ("street"), or "Pl." for "Platz" ("place"). We construct gazetteers for companies, cities and towns, streets, transport stops, and transit routes, from the knowledge resources described in Section 3.2. All gazetteers store name variants, database identifier information, as well as geo-locations.

We perform an entity linking step next to disambiguate the candidate entities of a recognized concept. Since our system is based on a very extensive set of company and geo-location entities, entity linking is particularly challenging. For example, many public transit route names are identical across German cities (e.g. "S1" for "train line #1"), and street names are also very often re-used in different cities. We implement a straight-forward, geo-location-based disambiguation algorithm. For ambiguous entities, such as streets, transit stops, or small villages, the algorithm chooses the candidate whose coordinates are contained in the geo-shape of "larger" entities co-occurring in the text. In turn, unambiguous stop or street entities can also be used to resolve "larger" ambiguous entities, e.g. transit routes, using a similar strategy. Additionally, Twitter allows users to tag locations in a message. This can either be a precise location (longitude, latitude) or a general location label (e.g., a city name with geo-shape). For tweets with a location label, we retain only candidate entities located within the user-tagged geo-shape, and then apply our disambiguation algorithm.

Finally, our pipeline detects events and event arguments by matching dependency parse trees of sentences to a set of automatically learned dependency patterns, as proposed by Xu et al. (2007).

Name	Arguments
Accident	Road, route, loc, time
Delay	Road, route, flight, cause, loc, time
Disaster	Type, trigger, casualties, loc, time
Traffic Jam	Road, loc, time
Rail Replacem.	Route, loc, time
Road Closure	Road, cause, loc, time
Acquisition	Buyer, acquired, loc, time
Merger	Old, new, trigger, loc, time
Spin-off	Parent, child, loc, time
Layoffs	Company, trigger, number, loc, time
Strike	Company, trigger, loc, time
Insolvency	Company, trigger, cause, loc, time

Table 3: Event types recognized by our pipeline, and their arguments.

[9] avro.apache.org
[10] flink.apache.org
[11] github.com/kohlschutter/boilerpipe

We define an event in the spirit of the ACE / ERE guidelines (Linguistic Data Consortium, 2015) as a n-ary relation with a set of required and optional arguments, which include location and time. The dependency patterns we use for event detection are extracted from a manually curated set of event-specific training sentences, which were annotated with event type, argument types, and argument roles. For Twitter texts, where we typically cannot expect high-quality dependency parses, we complement our approach with a key phrase-based event detection strategy. By carefully selecting the trigger key phrase set, we can identify event types with high precision. Table 3 summarizes the events and arguments recognized by our system.

5 Web Application

Annotated documents are persisted to a distributed Solr index[12]. The index stores basic document information, such as URL, title and text, and additionally information about extracted entities and events. It serves as a back-end for the visualization component of the system, which is realized as a light-weight web application. The front-end is available at `http://ta.dfki.de`.

Figure 2a shows the main view of the front-end. The view displays the most recent events on an interactive map. Each icon represents a single event. Events co-located in a particular area are clustered to avoid cluttering. The clusters split into individual event icons when zooming in on the location. The map is updated frequently to refresh the event set, giving a dynamic view of ongoing developments. The user interface provides various filtering options for event types, data sources, and transport modes, as well as a company search.

Events can be selected, which opens an overlay with detailed information about the event and its source document. Details include the event type, time, and location, as well as an explanatory text snippet from the source document. The detail view also shows a set of illustrative supply chains between companies that may potentially be affected by the event, see Figure 2b. Since data about suppliers is typically non-public information, we dynamically generate example supply chains by randomly selecting a pair of companies in the vicinity of the event's location, and computing a route between them using Mapbox [13].

[12] `lucene.apache.org/solr`

[13] `mapbox.com/api-documentation`

Selecting a "supply chain problem" involving a particular company opens another detail view. This detail view displays information about the affected company, and includes infobox-style facts from background knowledge bases, but also recent events and news referencing the company.

6 Evaluation

We conducted a preliminary evaluation of our system's event extraction performance on the dataset described in Table 1. Figure 3 shows the distribution of recognized events. The largest proportion of events is extracted from Twitter messages. Company-related events are mostly found in news articles, while mobility-related events are mainly reported in Twitter and RSS feeds.

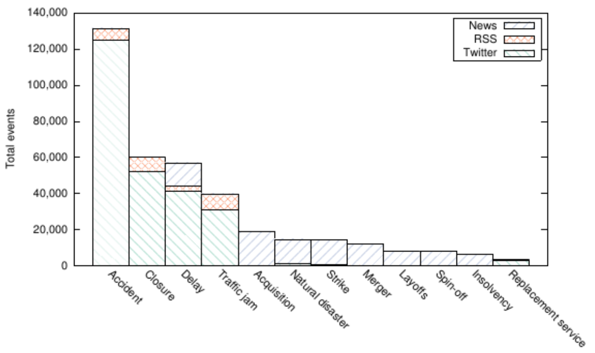
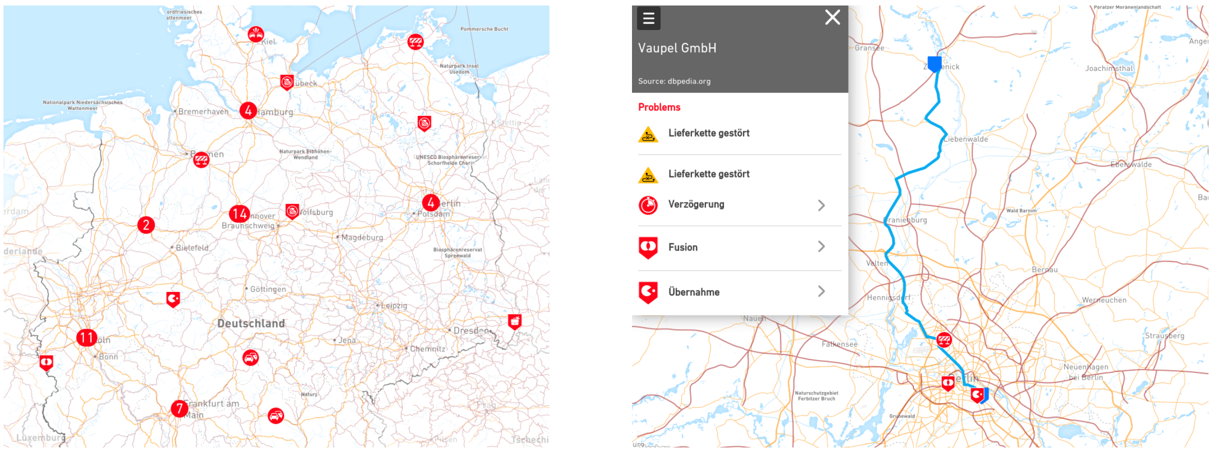


Figure 3: Event distribution for RSS, Twitter and News.

To evaluate event extraction accuracy, we manually judge a random sample of 150 documents (50 per source) for each event. A document is considered to correctly state an event if it explicitly reports a past, current or future event. All other documents are labeled as incorrect.

Table 4 lists the precision scores per source and micro-averaged across sources for selected event types. On average, 64% of the identified events are judged to be correct. Best results are observed for RSS feeds. This is an expected result, since traffic information RSS feeds are often well structured and employ very formalized wording. Some events types are more reliably observed in specific sources, e.g. *Strike* and *Layoffs* in news. The overall precision on Twitter is surprisingly high, with the exception of *Traffic Jam* events. This can be attributed to the fact that our key phrase-based approach used the German word "Stau" ("jam"), which often appears in non-traffic contexts to denote slow or halting progress.



(a) Event visualization UI

(b) Company view with supply chain route

Figure 2: User interface of the *Spree* system

Event type	Twitter	RSS	News	Avg
Traffic jam	0.28	1.0	–	0.64
Strike	0.58	–	0.66	0.62
Delays	0.74	0.94	0.26	0.65
Disaster	0.52	0.94	0.48	0.57
Layoffs	0.66	–	0.76	0.71

Table 4: Precision of event recognition for selected event types. Empty cells indicate that the corresponding event did not occur in the given document type.

7 Conclusion

We have presented *Spree*, a scalable, real-time event extraction system for mobility and industry events. Our system discovers and visualizes events from heterogeneous data streams, including Twitter, RSS feeds and news documents, on an interactive map. Data streams are automatically filtered for relevant events, and geo-located using information from the extracted entities. Our system recognizes an extensive set of fine-grained entities of different types, including companies, streets, routes and transport stops, and extracts ACE/ERE-style events, together with their argument fillers, using a dependency pattern-based approach. Implemented in Apache Flink, it is highly scalable and allows both batch and streaming processing.

Acknowledgments

This research was partially supported by the German Federal Ministry of Economics and Energy (BMWi) through the projects SDW (01MD15010A) and SD4M (01MD15007B), and by the German Federal Ministry of Education and Research (BMBF) through the project BBDC (01IS14013E).

References

A. Alexandrov, R. Bergmann, S. Ewen, J. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. Sax, S. Schelter, Ma. Höger, K. Tzoumas, and D. Warneke. 2014. The Stratosphere Platform for Big Data Analytics. *The VLDB Journal*, 23(6):939–964.

B. Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proc. of COLING*, pages 89–97.

W. Drozdzynski, H. Krieger, J. Piskorski, U. Schäfer, and F. Xu. 2004. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 1:17–23.

D. Ferrucci and A. Lally. 2004. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3–4):327–348.

Linguistic Data Consortium. 2015. Rich ERE annotation guidelines overview. `http://cairo.lti.cs.cmu.edu/kbp/2015/event/summary_rich_ere_v4.1.pdf`.

M. Osborne, S. Moran, R. McCreadie, A. Von Lunen, M. Sykora, E. Cano, N. Ireson, C. Macdonald, I. Ounis, Y. He, T. Jackson, F. Ciravegna, and A. O'Brien. 2014. Real-time detection, tracking, and monitoring of automatically discovered events in social media. In *Proc. of ACL: System Demonstrations*, pages 37–42.

Feiyu Xu, Hans Uszkoreit, and Hong Li. 2007. A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity. In *Proc. of ACL*, pages 584–591.