

Streaming Text Analytics for Real-Time Event Recognition

Philippe Thomas, Johannes Kirschnick, Leonhard Hennig, Renlong Ai,
Sven Schmeier, Holmer Hensen, Feiyu Xu, Hans Uszkoreit

Deutsches Forschungszentrum für Künstliche Intelligenz, Germany

firstname.lastname@dfki.de

Abstract

A huge body of continuously growing written knowledge is available on the web in the form of social media posts, RSS feeds, and news articles. Real-time information extraction from such high velocity, high volume text streams requires scalable, distributed natural language processing pipelines. We introduce such a system for fine-grained event recognition within the big data framework Flink, and demonstrate its capabilities for extracting and geo-locating mobility- and industry-related events from heterogeneous text sources. Performance analyses conducted on several large datasets show that our system achieves high throughput and maintains low latency, which is crucial when events need to be detected and acted upon in real-time. We also present promising experimental results for the event extraction component of our system, which recognizes a novel set of event types. The demo system is available at <http://dfki.de/sd4m-sta-demo/>.

1 Introduction

In the last few years text analytics has assumed a very important role in the area of large scale data processing, in particular, big data analytics of unstructured textual data. Its applications range from fulfilling very specific information needs to building up knowledge resources serving a variety of purposes (e.g. social media monitoring, business intelligence, and knowledge organization). An important task of text analytics is detection and monitoring of events reported in various texts. For example, mobility providers may wish to better serve their customers by incorporating real-time infor-

mation about short-term disruptions across the general mobility infrastructure (e.g. accidents, delays, roadblocks), whereas supply chain managers require timely information about critical supplier events (e.g. liquidity problems, strikes, disasters).

While general text analytics pipelines for event detection focus on detecting and tracking “news topics”, such as earthquakes or crisis events (Allan et al., 1998; Osborne et al., 2014), monitoring fine-grained events, such as strikes at a particular facility of a company, or a traffic accident on a specific road crossing, raises additional challenges. It requires more extensive linguistic analysis of texts, including named entity recognition (NER) for non-standard entity types such as roads or facility locations, entity linking (EL) (Dredze et al., 2010) and relation extraction (RE) (Mintz et al., 2009). Furthermore, much larger and more domain-specific knowledge bases for NER and EL have to be incorporated, as opposed to existing systems (Ji et al., 2014). For real world applications, it is also necessary to combine information extracted from unstructured texts with (semi-)structured information provided by domain databases such as public transport timetables, and traffic information systems.

The ever-increasing volume and velocity of textual data available on the web raises additional challenges from an engineering point of view, especially if we aim for real-time processing. Data must be processed and transformed “on the fly” so that, when it reaches a persistent data store, it is immediately available for querying and interpretation (Orenstein et al., 2015). The system also needs to be able to handle the burstiness of input data streams, i.e. it must be capable of adapting to the high variability in input volume and handle the resulting back-pressure appropriately. Existing NLP pipeline frameworks, such as UIMA (Ferrucci and Lally, 2004), often only allow for some

degree of parallelization, but are not designed to handle big data streams.

In this paper, we introduce a scalable, distributed linguistic analysis pipeline to detect fine-grained events from heterogeneous text sources. The pipeline’s main functionalities include entity recognition, entity linking to well established knowledge bases and event recognition. To ensure scalability, robustness, and near real-time distributed processing, our platform is implemented within the big data analytics framework Flink (Alexandrov et al., 2014). Our contributions in this paper are as follows:

- A linguistic analysis pipeline for detecting and localizing events, including NER for fine-grained and non-standard entity types, and a novel entity linking approach for resolving highly ambiguous geo-entities such as street or train station names
- A scalable, distributed architecture for fault-tolerant processing of data streams in (near) real-time and a performance analysis in terms of throughput and scalability
- Experimental results on the event extraction quality of the pipeline, based on a dataset collected over a 3 month period

Moreover, we will discuss some observations on the performance of our NER, EL and RE approaches (see Section 4).

2 NLP Pipeline

We give an overview of the pipeline architecture, which is shown in Figure 1. The current system is designed to deal with various text types such as Twitter messages, RSS feeds, and web pages, and simultaneously handles German- and English-language documents. For each resource type, an example is shown in Figure 2.

2.1 Preprocessing

Preprocessing consists of the following steps: boilerplate detection, language detection, sentence segmentation and tokenization, POS tagging and dependency parsing.

Due to the high heterogeneity of website contents, processing HTML websites is an inherent challenge. Boilerplate detection is used to retain the main content of an HTML document (Kohlschütter et al., 2010), `langid.py` (Lui and Baldwin, 2012) for language detection –

Type	Size	Resource	Examples
Company	112,347	Internal	BMW, Bayer AG
City	27,075	OSM	Berlin, Hof
Street	104,598	OSM	Hauptstrasse, A1
Station	9,860	Dt. Bahn	S+U Hauptbahnhof
Route	25,907	Dt. Bahn	ICE 557, U2, M47

Table 1: Named entity types and size of the corresponding gazetteers recognized by our pipeline. *Route* refers to public transport identifiers.

which achieves high accuracy for various text types and short documents. In our current implementation, we focus on English and German documents. POS tagging and dependency parsing is performed by the Mate Tools suite (Bohnet, 2010) for German, and with Stanford CoreNLP (Manning et al., 2014) for English.

2.2 Named Entity Recognition

Table 1 represents the main entity types covered by our system, besides standard types such as date and time expressions. NER is realized by a combination of Stanford CoreNLP (Manning et al., 2014) and the general purpose entity recognition toolkit SProUT (Drozdowski et al., 2004). SProUT is a rule-based framework that unifies analyzed features of each module (e.g. tokenizer, morphology, and gazetteer) as typed feature structures and applies grammar rules based on regular expressions to recognize entities.

Besides some general rules for recognizing organizations and datetime expressions, we implemented particular rules to deal with frequent morphologic variations of names (e.g. German “Strasse” and “Straße” for “street”) and abbreviations (e.g. “Pl.” for “Platz” (“place”)), as well as special time or date formats often used in RSS feeds and news articles. We construct gazetteers for companies, cities, streets, public transport stations, and routes from existing knowledge resources. The gazetteers store information about naming variants, database identifiers, and geospatial shapes¹. For geographic entities (i.e., cities and streets) we utilize data from OpenStreetMap (OSM). The public transport datasets provided by Deutsche Bahn AG contain information and geo-shapes for public transport stations, timetables, and interconnecting routes within Germany. The company gazetteer is constructed from

¹Shapes consist of a set of geo-coordinates describing a polygon.

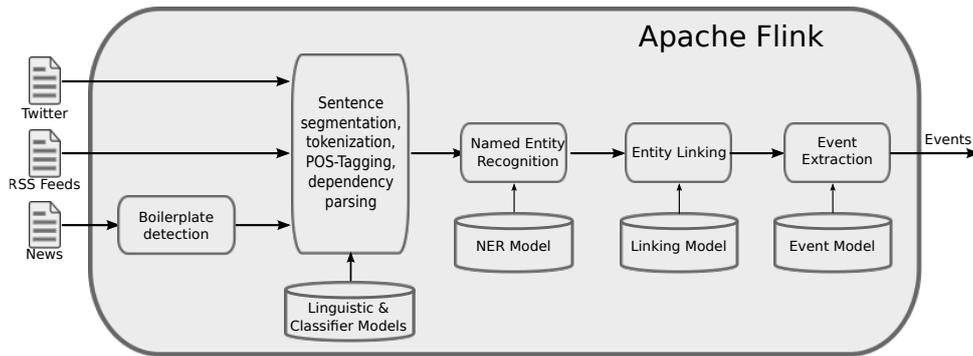


Figure 1: System Architecture

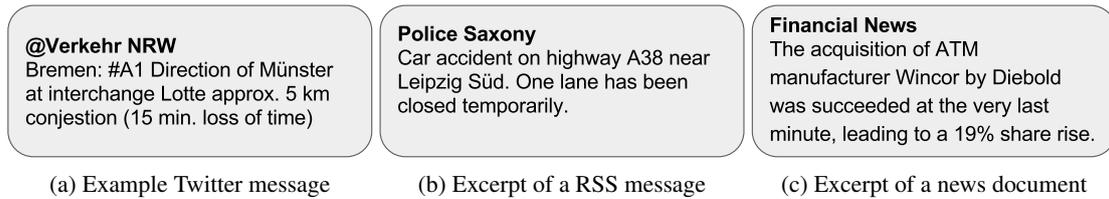


Figure 2: Examples for the three different data sources covered by our pipeline.

a large dataset of enterprises, which includes small and medium-sized enterprises, and thus considerably extends the data available e.g. in Wikipedia or Freebase (Bollacker et al., 2008).

2.3 Entity Linking

An entity linking step is required after NER to disambiguate the recognized candidate entities. Since our system utilizes a large set of company and geo-location entities, entity linking is particularly challenging. For example, many public transport route names are synonymous across German cities (e.g. "S1" for "suburban train line #1" exists in more than 15 metropolitan regions) and street names are also very often re-used in different municipalities (see Table 2).

We implemented a novel geo-location based disambiguation strategy. For ambiguous entities the algorithm chooses the candidate whose coordinates are contained or intersect with the geo-shape of "larger" entities co-occurring in the same text. For example, a street name is typically resolved to the correct database entry by testing if the street's shape is contained in the geo-shape of a city mentioned in the same document. In turn, unambiguous stop or street entities can also sometimes be used to disambiguate other entities, e.g. transit routes, using a similar strategy. In other words, we combine document context and geo-location context to link entity mentions to their

Streetname	# concepts
Hauptstraße	6,889
Schulstraße	5,190
Dorfstraße	5,138
Gartenstraße	4,918
Bahnhofstraße	4,761

Table 2: List of the five most frequently found street names in our dataset of German cities.

correct knowledge base entry.

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). In our corpus of more than 3.8 million tweets, we observe 2.7% to be tagged with a location shape. For tweets with a location label, we prioritize recognized entities within the user tagged region.

We manually identified more than 150 Twitter channels which are frequently used to communicate mobility related information. This includes channels from mobility providers, local radio stations, police departments, and community forums. Overall, approximately 31.21% of all collected tweets are sent over one of these channels. For each of these channels we manually identified their regions of main interest, usually a list of cities or a county. During linking, we use this information to remove entities located outside of the region of interest.

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

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

2.4 Event Detection

In this work, we define events in the spirit of the definitions of ACE (Automatic Content Extraction) guidelines (Doddington et al., 2004) as n-ary relations with a set of required and optional arguments, including location and time. For example, a *strike* event has a required argument *company*, one or multiple *location* arguments, and an optional argument *time* (see Table 3). We hence use a more fine-grained definition of what constitutes an event than the document-level view typically assumed in topic detection and tracking (Allan et al., 1998).

We detect events by matching dependency parse trees of sentences to relation-specific patterns, as described by Xu et al. (2007). The dependency patterns for each relation are extracted automatically from a set of 2,000 training documents, which have been manually annotated for event type, argument types, and roles. For Twitter, we additionally implement a keyword-based event detection strategy, since dependency parsing is likely to often produce erroneous results given the very informal language of many tweets. We define a set of relation-specific trigger phrases to detect events, which are matched both to hashtags and general tweet text. By carefully selecting the trigger phrase set, we can identify events with high precision. Figure 3 shows a screenshot of our web demo for NER, EL and RE.

3 Stream Processing Architecture

Our system is separated into three distinct functions: document retrieval, processing and annotation, and data storage - which are connected via a distributed message queue system, Kafka².

²kafka.apache.org

Retrieval is handled by individual data source adapters that fetch a continuous stream of new documents and forward the results into the message bus. This relieves the annotation processor from source specific data handling as well as buffers documents in case of large traffic bursts. A similar, less versatile, message-based architecture, has been presented by Kirschnick and Thomas (2017) for NER in the biomedical domain.

The core processing itself is handled by Apache Flink, a distributed, streaming data flow engine that provides data distribution, communication, and fault tolerant stream computation. We modeled the NLP pipeline within Flink as a series of transformations, that each wrap one NLP aspect. Flink connects to the message bus and forwards each new document through the pipeline. Fault tolerance, using check-pointing, guarantees exactly once processing, avoiding to double process the same document in case of failures. Furthermore, with adaptive back pressure handling, Flink will back off from retrieving new documents when the pipeline itself falls behind in processing, which guarantees the fastest possible processing across all compute nodes.

Together, these frameworks 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.

Transformations subsequently enrich an internal document representation whose schema is inspired by the Common Analysis Structure implemented in UIMA. This schema defines major elements, such as sentences, tokens, concepts, relations and generic attributes, including provenance information, consisting of annotator, confidence and license information for traceability. Other annotations are either realized as attributes of these classes, or in a generic fashion using a labeled span scheme. The data schema is extensible and allows us to easily integrate additional annotation components, as well as multiple different annotators for the same annotation type (e.g. when using both Stanford and SProUT for NER). We use Avro³, a compact binary data format, to efficiently serialize documents between pipeline processing

³avro.apache.org

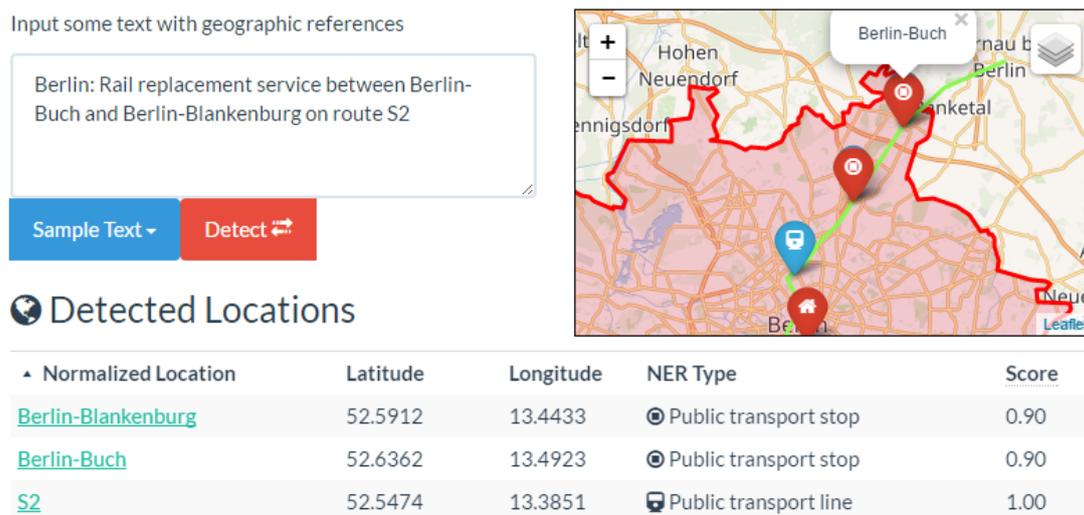


Figure 3: Screenshot of our web based demo system for location detection.

steps.

Finally, annotated documents are pushed back to the message system, where they can be read by multiple consumers. In our implementation we archive all results for offline analysis and index a rolling window of the results for immediate querying through a dashboard. The dashboard can be used to serve alerts and notifications of extracted events in near real-time, as they are happening.

4 Observations and Insights

In this section we share observations made during the development of our linguistic pipeline.

Preprocessing: During preprocessing we observed that boilerplate detection sometimes retains navigational text, which impacts the subsequent linguistic steps. Twitter and RSS feeds are easier to process due to the structured and well-defined format they are provided in (JSON and XML).

Named entity recognition: Our current approach substantially relies on gazetteers and grammatical rules implemented in SProUT. This approach provides some inherent difficulties for NER: First, we are unable to detect specific-type entities not covered by the gazetteers (since Stanford only detects ‘larger’ locations). To ameliorate this problem, we implemented a fuzzy matching strategy, but lower than desirable recall is still an issue. Second, cross-type ambiguity of e.g. public transport stop names sharing the name with the street or borough they are located in, remains a challenge. For example, “Baumschulenweg” can refer to a public transport train station in Berlin, a street in more than 15 cities,

or a locality in Berlin. Third, we observe a high ambiguity between smaller cities and German nouns, e.g. Regen (rain), Dom (cathedral), or Strom (stream/river, but also electricity). In this work we utilize a blacklist of frequent terms to identify and remove such entities. As an alternative to SProUT, we plan to incorporate conditional random field models (Lafferty et al., 2001), trained for non-standard entity types in the future (Schwarzenberg et al., 2017).

Entity linking: Entity linking is implemented using a geo-location based disambiguation strategy. This strategy works best for documents, mentioning the larger area (e.g. a city) and smaller locations contained within this area in the same sentence. In cases where multiple cities match a named entity, we require a popularity measure to rank target concepts. A simple measure currently implemented is the surface area of a shape. However, this measure can sometimes be misleading in case of relatively large but sparsely populated places (e.g. Frankfurt (Oder) is only 40% smaller than Frankfurt (Main), but with only 8% of the inhabitants). A potentially helpful measure is the number of inhabitants in a specific area, which is only provided for approximately 4.6% of all areas contained in OSM.

Event extraction: Event extraction uses relation-specific dependency patterns for RSS-feeds and news articles. For Twitter we define a set of relation-specific trigger phrases. The latter event detection strategy poses a problem for metaphoric word usage. For instance, “a landslide victory in Berlin” would be tagged as nat-

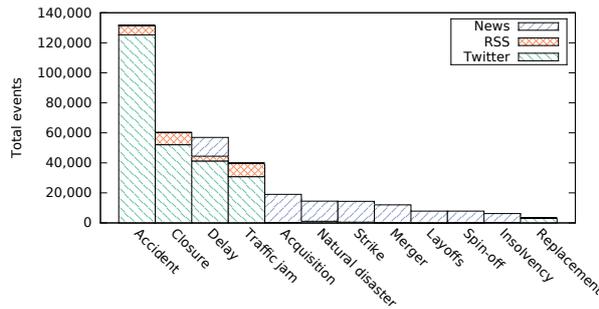


Figure 4: Distribution of extracted events for the three different text sources.

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.

ural disaster. Currently, the event extraction module recognizes events and arguments according to the types defined in Table 3. In the next step, we would like to aggregate a detected event into a geospatial representation. For instance, a traffic jam event on the motorway A9 between Munich and Eching should be reduced to this specific section, and not the full length of the highway.

5 Relation Extraction Experiments

We applied our pipeline to a dataset of 3,789,803 tweets, 412,652 RSS feeds, and 860,307 news documents collected in the time period of Jan 1st, 2016 to March 31st, 2016. Figure 4 visualizes the distribution of the 12 different event types across all sources. The largest proportion of events is extracted from Twitter messages, which also constitutes the majority of all processed documents. Company related events are mostly extracted from news documents, while mobility related events are almost exclusively detected in Twitter and RSS feeds.

For each event type, we manually judge the correctness of a random sample of 50 documents per source (if available for a given relation). Documents were considered to *correctly* state an event if their text explicitly reported the event, regard-

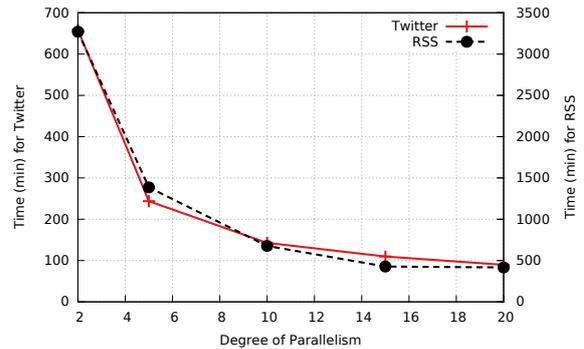


Figure 5: Experiment runtime with varying degrees of parallelism on the testing corpus.

less of whether it was ongoing, took place in the past, or was announced for the future (e.g. in the case of strikes). Events that were only implied were labeled as *incorrect*. Table 4 lists the precision scores of a subset of the events, for all three sources and micro-averaged across sources.

Best results are generally observed for RSS feeds. This is an expected result, as we selectively collect only RSS feeds from traffic information sources. Some event types are more reliably observed in specific sources, e.g. *Strike* and *Layoffs* in news. The overall accuracy of event recognition on Twitter is surprisingly high, with the exception of *Traffic Jam* events. This, however, can be attributed to the fact that the keyphrase pattern approach employed for Twitter used the German word “Stau” (“jam”), which is often used in other, non-traffic contexts to denote slow or halting progress. On average, 64% of the identified events are judged to be correct.

6 Performance and Scalability

To assess the performance of our system we selected 2,875,000 Tweets and 3,093,456 RSS messages, corresponding to a weekly data sample.

We first measured the time that each individual step of the processing pipeline takes, to assess the latency that an ideal streaming pipeline would possess. This determines the delay between message acquisition and availability for querying or inspection of downstream systems. Overall, RSS documents contain more text, and thus incur higher processing costs, compared to the much shorter Twitter messages. On average, our pipeline takes roughly 1s to process a RSS document and 250ms for a Twitter message.

We are also interested in the scaling properties

of distributing the processing across multiple machines. We observed that the volume of messages is not constant across each source, instead these sources show a bursty behavior. Large events, for example, can generate huge temporary spikes in message volume – but at the same time we would like to retain the ability to process all messages in near real-time by scaling out. The experiment setup consisted of a cluster of 4 machines (32 logical CPUs and 256 GB RAM each), running Ubuntu 14.04 and Flink v1.1.2. We varied the degree of parallelism by specifying the number of task managers that Flink could utilize to distribute the processing and measured the overall runtime to process each dataset sample.

Figure 5 shows the results of the scaling experiment, varying the parallelisms from 2 to 20. For both datasets the processing time drops significantly when more and more tasks managers are added. The performance gains are smaller for larger degrees of parallelism, due to a corresponding increase in communication costs between worker nodes. In the highest setting, we measured a throughput of 530 docs/sec and 123 docs/sec for Twitter and RSS respectively.

The price for the gained throughput is a modest processing overhead introduced by Flink itself as well as the time it takes to transport the documents over the network. With 20 task managers we observed an overall processing overhead of 10%.

7 Related Work

The ReDites real-time event detection and tracking system presented by Osborne et al. (2014) focuses on events from the security domain, and uses Twitter as its sole data source. As our system, it is fully automatic, geo-locates events, and supports multiple languages. Its scope, however, does not encompass deep linguistic analyses of input documents for NER, EL and RE, since it uses a shallow definition of event detection in the spirit of Topic Detection and Tracking (TDT). The system can therefore not be used e.g. for knowledge base population. Other TDT-like systems include Trend Miner (Preojuic-Pietro and Cohn, 2013) and the Social Sensor system (Aiello et al., 2013).

Most similar to our system is the approach presented by Tanev et al. (2008). Their system also extracts ACE-style events together with their arguments from news texts. They utilize, in their own words, a ‘linguistically poor’ approach that

relies on simple 1 and 2-slot extraction patterns that are acquired by bootstrapping. The system focuses on security-related events, employs TDT-style clustering techniques, but for efficiency reasons processes only the first sentence of each document. Our system scales to full documents, and can hence detect a much larger set of events from each document. Piskorski and Atkinson (2011) later extend the approach of Tanev et al. to more fine-grained event schemas, and to support event detection in 7 languages.

8 Conclusion

In this work we introduced a system for scalable, realtime event extraction. Our system currently implements both German and English analysis pipelines that include components for NER, EL and RE. It processes different input text data streams, including high-volume, high-velocity sources such as Twitter, but also continuous crawls of web documents and RSS feeds. A performance analysis conducted on several large datasets shows that processing large volumes in near-realtime is possible when running on the distributed stream processing platform Flink – not only achieving high throughput, but also maintaining low latency, crucial when extracted events need to be monitored and acted upon. The demo is available at <http://dfki.de/sd4m-sta-demo/>.

We find that linking fine-grained geo-entities (such as street or route names) is especially challenging, and will require better models. Furthermore, in future work we would like to extend our relation extraction component to handle more relations and event types, and to correctly distinguish between factual and future (or rumored) events, such as for example planned road works, or company acquisition offers.

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

- L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Gker, I. Kompatsiaris, and A. Jaimes. 2013. Sensing Trending Topics in Twitter. *IEEE Transactions on Multimedia* 15(6):1268–1282.
- 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).
- J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. 1998. Topic detection and tracking pilot study: Final report. In *Proc. of DARPA Broadcast News Transcription and Understanding Workshop*.
- B. Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proc. of COLING*. pages 89–97.
- K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proc. of SIGMOD*. pages 1247–1250.
- G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. 2004. The Automatic Content Extraction (ACE) Program - Tasks, Data, and Evaluation. In *Proc. of LREC*.
- M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. 2010. Entity disambiguation for knowledge base population. In *Proc. of COLING 2010*.
- W. Drozdzyński, 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.
- H. Ji, J. Nothman, and B. Hachey. 2014. Overview of TAC-KBP2014 Entity Discovery and Linking Tasks. In *Proc. of the Text Analysis Conference*.
- J. Kirschnick and P. Thomas. 2017. SIA: Scalable Interoperable Annotation Server. In *Proc. of BeCalm*. pages 138–145.
- C. Kohlschütter, P. Fankhauser, W. Nejdl, C. Kohlschütter, P. Fankhauser, and W. Nejdl. 2010. Boilerplate Detection Using Shallow Text Features. In *Proc. of WSDM*. pages 441–450.
- J. D. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML*. pages 282–289.
- M. Lui and T. Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proc. of ACL: System Demonstrations*. pages 25–30.
- C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proc. of ACL: System Demonstrations*. pages 55–60.
- M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In *Proc. of ACL-IJCNLP*. pages 1003–1011.
- G. Orenstein, C. Doherty, K. White, and S. Camiña. 2015. *Building Real-Time Data Pipelines*. O’Reilly.
- 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*.
- J. Piskorski and M. Atkinson. 2011. Frontex real-time news event extraction framework. In *Proc. of KDD*. KDD ’11, pages 749–752.
- D. Preoțiuc-Pietro and T. Cohn. 2013. A temporal model of text periodicities using gaussian processes. In *Proc. of EMNLP*. pages 977–988.
- R. Schwarzenberg, L. Hennig, and H. Hensen. 2017. In-Memory Distributed Training of Linear-Chain Conditional Random Fields, with an Application to Fine-Grained Named Entity Recognition. In *Proc. of GSCL*.
- H. Tanev, J. Piskorski, and M. Atkinson. 2008. Real-time news event extraction for global crisis monitoring. In *Proc. of NLDB 2008*. pages 207–218.
- F. Xu, H. Uszkoreit, and H. Li. 2007. A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity. In *Proc. of ACL*. pages 584–591.