EVELIN:
Exploration of Event and Entity Links in Implicit Networks

Andreas Spitz
spitz@informatik.uni-heidelberg.de

Satya Almasian
almasian@stud.uni-heidelberg.de

Michael Gertz
gertz@informatik.uni-heidelberg.de

Institute of Computer Science, Heidelberg University
Im Neuenheimer Feld 205, 69120 Heidelberg, Germany

ABSTRACT

Implicit networks that describe latent entity relations have been demonstrated to be valuable tools in information retrieval, knowledge extraction, and search in document collections. While such implicit relations offer less insight into the types of connection between entities than traditional knowledge bases, they are much easier to extract from unstructured textual sources. Furthermore, they allow the derivation of relationship strength between entities that can be used to identify and leverage important co-mentions, based on which complex constructs of semantically related entities can be assembled with ease. One example of such implicit networks are LOAD graphs, which encode the textual proximity of location-, organization-, actor-, and date-mentions in document collections for the exploration, identification and summarization of events and entity relations. Here, we present EVELIN as a graphical, web-based interface for the exploration of such implicit networks of entities on the example of a large-scale network constructed from the English Wikipedia. The interface is available for online use at `http://evelin.ifi.uni-heidelberg.de/`.

Keywords

Entity network; latent network; event extraction; search; summarization; knowledge graph; entity ranking

1. INTRODUCTION

Knowledge bases are invaluable sources of structured, discrete relations between entities that have numerous applications in information retrieval and natural language processing. Tasks such as search, disambiguation or question answering can easily be augmented by structured entity relations stored in knowledge bases such as YAGO, DBpedia, Wikidata, or the Google Knowledge Graph. However, the practical applicability of such resources depends on the existence of structured knowledge for the domain under investigation. In cases where this information is unavailable, implicit entity networks that can be extracted directly from unstructured text have recently been shown to offer valuable support in information retrieval from large document collections [16]. Based on a weighted network structure that is extracted from entity co-occurrences in unstructured text, these methods enable the use of established information retrieval approaches on a graph representation of the text (for an early example of such an adaptation, see [12]).

As a result, implicit network approaches offer an important augmentation that the inclusion of knowledge bases alone cannot provide. While the integration of knowledge bases in search and retrieval tasks introduces additional, *external* knowledge, implicit networks induce a graph structure exclusively from the *intrinsic* information that is contained in the documents. Thus, implicit networks are applicable to arbitrary corpora of unstructured text, where they support exploratory measures. In combination with knowledge bases and entity linking techniques, implicit networks of entities can serve not only as indices and data structures for the occurrence of entity mentions in the underlying document collection, but also aid in the construction and exploration of complex entity structures that directly enable the identification of events in which these entities participate. Therefore, implicit networks stand to augment search applications in document collections when entities are both the input as well as the output of queries, and can be used in summarization as well as provenance detection. In this paper, we present EVELIN as a comprehensive system that utilizes such entity networks for search, exploration and ad hoc summarization on an underlying document collection.

Contributions. We provide EVELIN as an online web interface for searching entities and events in implicit entity networks, thus offering much of the functionality of knowledge graphs on unstructured document collections. To the best of our knowledge, it is the first holistic system that takes full advantage of the versatility of implicit entity networks and allows entity exploration, summarization, and provenance retrieval. Furthermore, it enables the composition of complex entity structures for event detection and summarization. Since the underlying graph can be constructed for arbitrary document collections, EVELIN is not limited to one specific corpus. We provide the system online as a web service[1] that runs on the implicit network extracted from the textual content of the English Wikipedia, with entity links to Wikidata as a knowledge base.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3054721

[1] EVELIN is available online, along with a user's manual: `http://evelin.ifi.uni-heidelberg.de`

2. RELATED WORK

Over the last years, a number of applications have been proposed that cover similar use cases. One of the first such systems is Broccoli, which extends SPARQL queries with textual co-occurrences [4], thus combining full text with knowledge base search on Wikipedia. With a focus on news articles, STICS uses entity auto-completion to suggest related entities for queries [8], but does not include temporal information beyond the publication date. The system has seen a couple of updates, including the visualization of trends in entity occurrences [7]. More recently, it was updated to include weighted co-occurrences of entity n-tuples for auto-completion suggestions [13]. Several other systems also focus on search and visualization of streams or collections of news articles, often in relation to entities in a knowledge base. Exposé introduces a time-aware retrieval approach for organizing news articles and linking them to Wikipedia as an event repository [11]. The approach is mostly focused on a temporal order but includes entities as facets for filtering the results. In a similar approach, Contextualizer serves as an interface for browsing news documents, by retrieving candidate documents based on user-selected keywords that are then ranked by contextual similarity and temporal aspects and linked to external Wikipedia information as a context [19]. Geared towards large-scale aggregation of news events, Event Registry is focussed on news articles, named entity annotation and date identification [9]. To this end, articles are clustered by contained named entities and content to obtain aggregate events. NewsStand offers a geographic event-centric visualization and exploration of news topics, including spatial keyword distributions [18]. It has been extended by a large number of additional aspects such as CrimeStand or brands. With a similar focus on the geographic aspect, Frankenplace is designed as a search interface for interactive thematic maps [1]. It provides geographic context for queries to an underlying document collection and utilizes topic modelling to leverage both themes and locations as dimensions for the exploration to search results. Expedition is a system with a focus on scholarly search [14], which filters and refines documents interactively on a timeline based on entity selections. Similarly, DigitalHistorian is a stand-alone tool aimed at temporal and entity-centric corpus exploration in the Digital Humanities [6]. InfoScout allows augmented, subject-centric investigation and is thus focussed on person mentions [10]. Based on an underlying knowledge base, XKnowSearch infuses traditional or multilingual keyword searches with structured information by introducing an intermediate query entity graph layer that maps keywords to entities, which are then used in the search [20]. Similarly, SemFacet combines search capabilities with knowledge base support for faceted search using document meta data [3].

Two recent approaches are most closely related to the work presented here. Ceroni et al. introduce a system for detecting and pinpointing events in document collections based on the co-occurrences of named entities [5], but focus on the task of validating event occurrences in the collection. The Entity Relatedness Graph is designed for the exploration of entity relations [2]. It uses the concept of entity similarity for ranking adjacent entities, but unlike our approach relies on the Wikipedia link structure for learning semantic relations between entities instead of using co-occurrences inside the document collection.

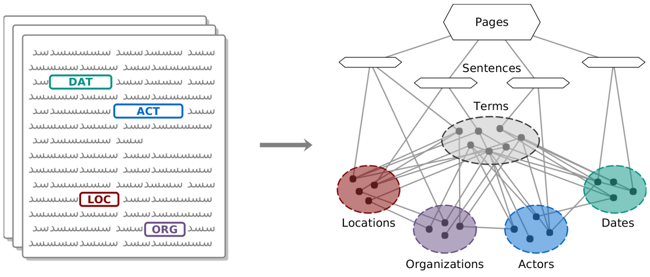


Figure 1: Schematic view of the LOAD graph extraction process from a document collection.

3. ENTITY GRAPH AND QUERY MODEL

In the following, we give a brief overview of the underlying data model and introduce the different algorithms that are used to retrieve information from the graph.

Data Model. EVELIN is designed as an interface for queries to the LOAD model, which is an entity co-occurrence graph representation of large document collections that was originally presented for event detection in unstructured texts [16]. We introduce the model only briefly and instead focus on the improved query functions. In the following, the neighbourhood $N(v)$ denotes the set of all nodes in the graph that are connected to a given node v.

The LOAD graph is a representation of the co-occurrences of the named entity types locations L, organizations O, actors or persons A, and dates D in the text. Remaining non-stop words are included as terms T, along with the containing sentences S and pages P (for a schematic overview, see Figure 1). Connectivity information for sentences and pages is binary, i.e., a sentence or page either contains an entity or it does not. In contrast, weights ω of edges between entities $x \in X$ and $y \in Y$ are derived from the textual distances of their mentions as

$$\omega(x,y) := \left(\log \frac{|Y|}{|N(x) \cap Y|}\right) \sum_{i \in I} \exp(-\delta_i(x,y))$$

where $\delta_i(x,y)$ denotes the distance in sentences between the occurrence of x and y in some instance i, and I is the set of all such co-occurrence instances (for the full derivation, see [16]). The weights that are generated in this way encode a directed importance of one entity for another. The LOAD graph can thus be used to answer queries about meaningful entity co-occurrences in the underlying document collection. By identifying events as composite subgraphs that consist of the participating entities, this ultimately allows the formulation of queries for events.

A query $\langle X|Q,n\rangle$ for retrieving information from the graph then consists of a target set $X \in \{L,O,A,D,T,S,P\}$, an integer n specifying the number of entities to retrieve from X, and a set of query entities $Q \subseteq L \cup O \cup A \cup D \cup T$. To answer a query, the aim is then to order all entities in X based on some ranking r that evaluates the importance of their relations to query entities $q \in Q$ by using the graph structure. The answer to a query is a set $X_n \subseteq X$ of the n top-ranked entities in X such that $r(x_n) > r(x)\ \forall x_n \in X_n, x \in X \setminus X_n$. In the following, we briefly describe the ranking approaches for the different target sets.

Entity Ranking ($X \in \{L,O,A,D,T\}$). For entities as target sets, we can distinguish two different scenarios. First, if the set of query entities contains only a single entity q

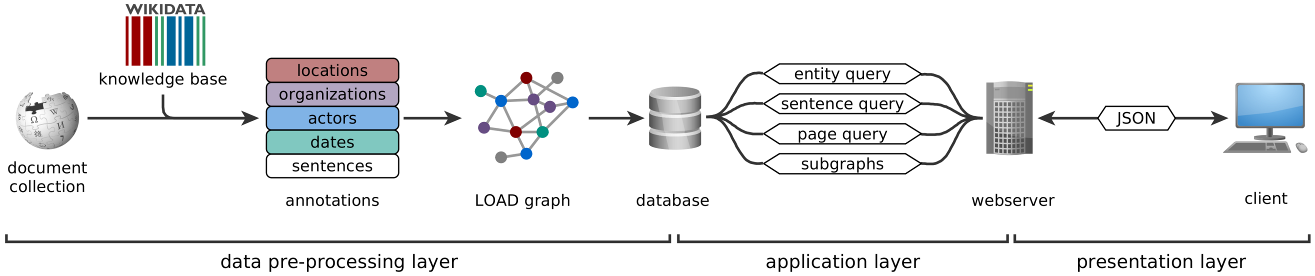


Figure 2: Schematic view of the data processing pipeline and system architecture. We use Wikipedia as textual input data for this demonstration, but the process can be applied to any document collection.

(i.e., $|Q| = 1$), then we rank entities in X by the weights of edges starting at q in the graph and let $r(x) = \omega(q, x)$. If q and x are not connected, then we simply set $r(x) = 0$. This ranking directly retrieves the most important entities x in the neighbourhood of entity q. To obtain a score in the interval $[0, 1]$, we normalize with the maximum observed score ω_{max} to any entity in the result set

$$\omega_{max} := \max_{x \in N(q)} \omega(q, x).$$

Second, if we have multiple query entities (i.e., $|Q| > 1$), then we employ a two-tier ranking system, in which we decompose the ranking score into two components $r = c.s$. This allows us to rank by the first component c and break ties according to the second component s. Here, c denotes the *cohesion* of the subgraph that is induced by the query entities. Formally, we define c as the number of query entities that are connected to a target entity beyond the first, i.e., $c(x) := |N(x) \cap Q| - 1$. Thus, we have $c \in \{0, ..., |Q| - 1\}$, with higher values indicating a higher connectedness of a target entity to the query entities. For the second component, we simply use the normalized sum of edge weights

$$s(x) := \frac{1}{s_{max}} \sum_{q \in Q} \omega(q, x)$$

where s_{max} is the maximum obtained sum of scores such that $s \in [0, 1]$. For the resulting ranking score we can thus set $r := c + s$ and observe that $r \in [0, |Q|]$, with higher scores denoting a higher importance. The single entity query then corresponds to the special case $c = 0$.

Sentence Ranking ($X = S$). Obtaining a ranking for sentences is less direct than a ranking for entities, since the edge weights between entities and sentences are binary and there is no notion of weight to these edges. Therefore, we employ a slightly different two-component ranking scheme $r = c.s$. The first component is identical to the case of entity ranking and denotes the cohesion, i.e., c is the number of query entities that are contained in a sentence x. To obtain the second component, we consider the k most important terms for each query entity and assign to each sentence a score that indicates how many of these terms it contains. Formally, let T_Q be the union of the k most important terms for all query entities in Q. Then we obtain s as the fraction of important terms that are contained in the sentence

$$s(x) := \frac{|N(x) \cap T_Q|}{|T_Q|}$$

Similar to the case of entity rankings, the combined ranking score is then the sum of the individual scores c and s.

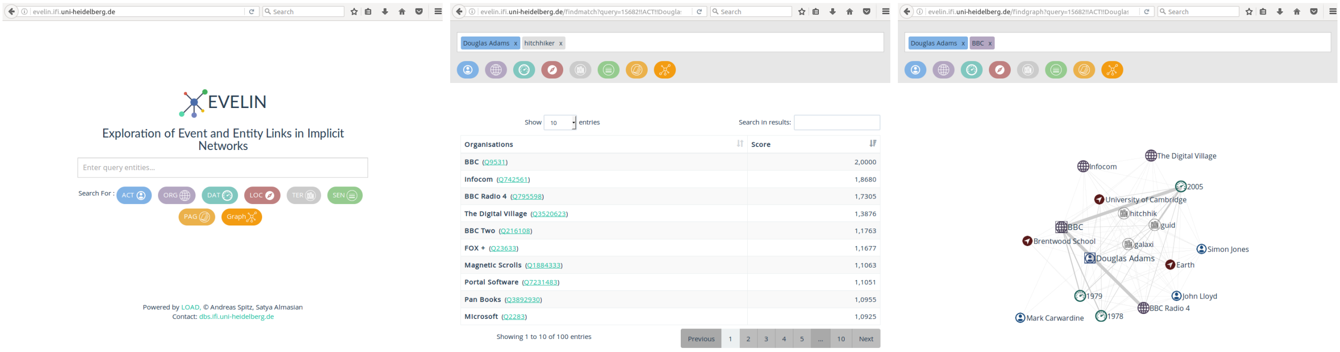
Page Ranking ($X = P$). For pages, a ranking can be obtained in almost the same way as for sentences. Here, we observe that each sentence belongs to exactly one page. Thus, we arrive at a ranking of pages by computing a ranking of sentences according to the query entities and then propagating the scores from the sentences to their respective pages. Using the same two component score as above, we define the cohesion of a page p as the maximum cohesion of any of its sentences $c(p) := \max c(x)$ for all $x \in p$. For the second component, we sum over the contributions of all individual sentences and obtain $s(p) = \sum_{x \in p} s(x)$. After normalizing with the maximum of all $s(p)$ and combining $r = c + s$, we again obtain a score $r \in [0, |Q|]$.

Subgraph Extraction. As an exploratory visualization tool, we also consider an experimental implementation of subgraph extraction. With the aim of highlighting the immediate neighbourhood of a given set of query entities, we first include all query entities in the subgraph. To discover additional nodes, we rely on entity queries as defined above. Specifically, we rank entities in each of the sets L, O, A, D and T according to their importance for the query entities. Then, we select the three highest ranked entities in each set that have a cohesion of $c \geq 1$ and include them in the graph. In a final step, we extract all edges between the selected nodes and include edge weights that can be used to visualize the importance of relations.

4. SYSTEM ARCHITECTURE

Based on the above methodology, we describe the system architecture for extracting the data and processing queries on the resulting entity network in the following. For an overview of the system architecture, see Figure 2.

Data Pre-processing. The extraction of a LOAD network is possible from any document collection, as long as named entities can be identified. To demonstrate the feasibility of the approach for large document collections and to provide comprehensive query choices, we use the unstructured text of all English Wikipdia articles (dump of May 1, 2016). This has the added benefit of allowing us to use Wikipedia links to identify named entities, which largely avoids the imprecision inherent to named entity recognition. The problem of only the first mention of an entity in a Wikipedia page being linked is corrected by a subsequent string search. We link discovered entities to Wikidata and classify them by directly classifying Wikidata entities (for further details on entity classification of Wikipedia links through Wikidata, see [15]). For the extraction of dates we use HeidelTime [17]. We construct a LOAD network from all discovered entities of types location, organization, actor and



evelin.ifi.uni-heidelberg.de
Search
EVELIN
Search For :
ACT
ORG
DAT
LOC
TER
SEN
PAG
Graph

Exploration of Event and Entity Links in Implicit Networks

Enter query entities...

Powered by LOAD, © Andreas Spitz, Satya Almasian
Contact: dbs.ifi.uni-heidelberg.de

evelin.ifi.uni-heidelberg.de/findmatch?query=15682!!ACT!!Dougla
Search
Douglas Adams x
hitchhiker x

Show 10 entries

Search in results:

Organisations	Score
BBC (Q9531)	2,0000
Infocom (Q742561)	1,8680
BBC Radio 4 (Q795598)	1,7305
The Digital Village (Q3520623)	1,3876
BBC Two (Q216108)	1,1763
FOX + (Q23633)	1,1677
Magnetic Scrolls (Q1884333)	1,1063
Portal Software (Q7231483)	1,1051
Pan Books (Q3892930)	1,0955
Microsoft (Q2283)	1,0925

Showing 1 to 10 of 100 entries

Figure 3: The EVELIN interface. Initial view with search bar and multiple query options (left), ranked results of a query with refinement options (center), and experimental subgraph visualization (right).

date with a maximum window size of 5 sentences by using the default LOAD implementation [16]. The resulting graph is constructed from 4.5M Wikipedia articles with 43.6M sentences that have at least one entity, containing 2.0M named entities, 5.2M distinct terms, and 1.3B edges.

Application Layer. An in-memory representation of a LOAD graph of the entire English Wikipedia is possible and allows extremely fast queries in the order of milliseconds. However, due to the high memory requirements (around 200GB for full efficiency), this is infeasible for a long-running, non-commercial application. Therefore, we use as demonstration server a Core i7 with 32GB main memory and an SSD drive. The network data is stored in a MongoDB, with separate collections for entities, terms, sentences, pages and edges. Entities are enriched with Wikidata information to obtain entity descriptions and canonical labels. A text index on the English canonical label is used for searching entities in the database by label and compiling a list of entity suggestions to the user based on input strings (an alternative solution is the use of prefix tries [7]). We rank entity suggestions by the text match score and break ties by the number of connected sentences in the network (i.e., by popularity). Graph edges are stored with precomputed weights. For edges that involve sentences, we use a collapsed storage format that contains both the sentence and the respective page of the entity or term. The query processing routines described in Section 3 are implemented in Java, and enable query processing speeds in the order of a few seconds or less for all but the experimental subgraph queries. To allow the application to serve queries from multiple users simultaneously, query processing is not parallel and allocates one thread per query. To avoid system overload in the case of multiple users that spam queries, we use an internal mapping of queries to browser fingerprints and limit the number of active queries per user.

Presentation Layer. The web interface is implemented via HTML and JavaScript and serves two primary purposes: classifying input terms and entities according to their entity type, and visualizing the output of ranked entity lists and subgraphs. For handling entity input and sending queries to the application layer, we use jQuery and pass entity information in both directions as JSON objects. The Bootstrap libraries[2] and Mustache web templates are used for the responsive layout and for displaying data tables. To recognize, classify and color input entities as they are entered, we use the tags-input and typeahead libraries of Bootstrap, which we extend by adding the required functionality for the color coding of entities. The interactive visualization of subgraphs is handled by the D3 JavaScript library[3], which uses a combination of HTML, CSS, and SVG to display data. Graphs are visualized with a force-directed layout. The web server itself is realized on top of the Java Spark micro framework[4] and is directly integrated with the application layer into a single application. Communication between user interface and server is built on AJAX and uses JSON for transmitting entity information in both directions (i.e, input query entities, output entities, and graph data). Examples of the interface, the search results, and a subgraph visualization are shown in Figure 3.

5. DEMONSTRATION SCENARIO

The demonstration is designed as a tour that organically guides the audience through the core functionality of the EVELIN interface and consists of three major stages. Alternatively, members of the audience have the option of interactively following the process and exploring the interface independently on their mobile devices.

Entity Exploration. Beginning with a single entity, such as a person or location, we demonstrate EVELIN's capability of discovering related entities. By adding newly discovered entities to a query or removing less interesting entities, we show how a single-entity query can grow dynamically into the exploration of an event or a person's timeline. We highlight how such event candidates of multiple query entities can be described and understood through term recommendations by the system.

Entity Summarization. Once an interesting entity combination is identified by the audience, we demonstrate the multi-entity summarization capability that is inherent to sentence queries. Based on the combination of entities that was discovered in the exploratory phase, we use EVELIN to retrieve a ranking of sentences from the entire corpus that best describes the selected entities as a composite.

Entity Linking. In a final step, we show how the system can be used to obtain provenance information for further studies, by effectively linking a set of entities to pages from the underlying document collection that provide the context of their co-occurrence. The entire demonstration process thus guides the audience from a single input entity

[2] `http://getbootstrap.com`

[3] `https://d3js.org`

[4] `http://sparkjava.com`

to a description and exploration of its relations to adjacent entities and the events that jointly describe them.

Subgraph Exploration. As an alternative, exploratory tool we also demonstrate the extraction of subgraphs around entities. While less precise than the entity-centric searches, a subgraph exploration can help the audience to obtain a first impression of an entity's relations and where to begin in a search. Thus, we use this aspect as an augmentation and visualization of the search process where appropriate.

Hardware requirements. The hardware requirements for the demonstration are minimal. Since the demonstration is realized via a web interface to a remote server application, the only requirement is a stable Internet connection. For the presentation of the interface, a laptop is entirely sufficient, although the presence of a larger screen or monitor enhances the experience for the audience, if available. Additionally, the audience is encouraged to explore the interface website on their own mobile devices, with a supplementary poster serving as a tutorial. In case of network unavailability, a downscaled graph that is constructed from a subset of Wikipedia pages will be available for offline demonstration.

6. CONCLUSION

With EVELIN, we presented a novel and comprehensive way of searching for entity-related information in unstructured, large-scale document collections that leverages the intrinsic relations of entities in the documents. Instead of relying on term-, entity-, or provenance data alone, we show how to combine all three data types organically in a single graph structure that can be used to navigate and explore the underlying document collection. As a result, we are able to retrieve entity rankings as well as descriptive sentences and pages, which allows us to induce and query a knowledge graph-like structure on otherwise unstructured document collections.

7. REFERENCES

[1] B. Adams, G. McKenzie, and M. Gahegan. Frankenplace: Interactive Thematic Mapping for Ad Hoc Exploratory Search. In *WWW*, 2015.

[2] N. Aggarwal, K. Asooja, H. Ziad, and P. Buitelaar. Who Are the American Vegans Related to Brad Pitt?: Exploring Related Entities. In *WWW Companion*, 2015.

[3] M. Arenas, B. Cuenca Grau, E. Kharlamov, S. Marciuska, D. Zheleznyakov, and E. Jimenez-Ruiz. SemFacet: Semantic Faceted Search Over YAGO. In *WWW Companion*, 2014.

[4] H. Bast, F. Bäurle, B. Buchhold, and E. Haußmann. Semantic Full-text Search with Broccoli. In *SIGIR*, 2014.

[5] A. Ceroni, U. Gadiraju, J. Matschke, S. Wingert, and M. Fisichella. Where the Event Lies: Predicting Event Occurrence in Textual Documents. In *SIGIR*, 2016.

[6] D. Gupta, J. Strötgen, and K. Berberich. DIGITALHISTORIAN: Search & Analytics Using Annotations. In *HistoInformatics Workshop at DH*, 2016.

[7] J. Hoffart, D. Milchevski, and G. Weikum. AESTHETICS: Analytics With Strings, Things, and Cats. In *CIKM*, 2014.

[8] J. Hoffart, D. Milchevski, and G. Weikum. STICS: Searching with Strings, Things, and Cats. In *SIGIR*, 2014.

[9] G. Leban, B. Fortuna, J. Brank, and M. Grobelnik. Event Registry: Learning About World Events From News. In *WWW Companion*, 2014.

[10] S. McKeown, M. Buivys, and L. Azzopardi. InfoScout: An Interactive, Entity Centric, Person Search Tool. In *SIGIR*, 2016.

[11] A. Mishra and K. Berberich. EXPOSÉ: Exploring Past News for Seminal Events. In *WWW Companion*, 2015.

[12] F. Rousseau and M. Vazirgiannis. Graph-of-word and TW-IDF: New Approach to Ad Hoc IR. In *CIKM*, 2013.

[13] A. Schmidt, J. Hoffart, D. Milchevski, and G. Weikum. Context-Sensitive Auto-Completion for Searching with Entities and Categories. In *SIGIR*, 2016.

[14] J. Singh, W. Nejdl, and A. Anand. Expedition: A Time-Aware Exploratory Search System Designed for Scholars. In *SIGIR*, 2016.

[15] A. Spitz, V. Dixit, L. Richter, M. Gertz, and J. Geiß. State of the Union: A Data Consumer's Perspective on Wikidata and Its Properties for the Classification and Resolution of Entities. In *Wiki Workshop at ICWSM*, 2016.

[16] A. Spitz and M. Gertz. Terms over LOAD: Leveraging Named Entities for Cross-Document Extraction and Summarization of Events. In *SIGIR*, 2016.

[17] J. Strötgen and M. Gertz. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation*, 2013.

[18] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A New View on News. In *SIGSPATIAL*, 2008.

[19] N. K. Tran, A. Ceroni, N. Kanhabua, and C. Niederée. Time-travel Translator: Automatically Contextualizing News Articles. In *WWW Companion*, 2015.

[20] L. Zhang, M. Färber, and A. Rettinger. XKnowSearch!: Exploiting Knowledge Bases for Entity-based Cross-lingual Information Retrieval. In *CIKM*, 2016.

Andreas Spitz is a PhD student in Computer Science at Heidelberg University, where he received his M.Sc. in Computer Science for the analysis of news networks in 2014. His research focusses on information retrieval, network analysis and natural language processing.

Satya Almasian is a Master student in Scientific Computing at Heidelberg University, with a focus on data mining and computational linguistics. She received her B.Sc. in software engineering and Computer Science at the Azad University of North Tehran, Iran, in 2014.

Michael Gertz is professor of Computer Science and head of the database systems research group at Heidelberg University since 2008. Previously, he was an associate professor of Computer Science at UC Davis, California.