Terms over LOAD: Leveraging Named Entities for Cross-Document Extraction and Summarization of Events

Andreas Spitz
spitz@informatik.uni-heidelberg.de

Michael Gertz
gertz@informatik.uni-heidelberg.de

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

ABSTRACT

Real world events, such as historic incidents, typically contain both spatial and temporal aspects and involve a specific group of persons. This is reflected in the descriptions of events in textual sources, which contain mentions of named entities and dates. Given a large collection of documents, however, such descriptions may be incomplete in a single document, or spread across multiple documents. In these cases, it is beneficial to leverage partial information about the entities that are involved in an event to extract missing information. In this paper, we introduce the LOAD model for cross-document event extraction in large-scale document collections. The graph-based model relies on co-occurrences of named entities belonging to the classes *locations*, *organizations*, *actors*, and *dates* and puts them in the context of surrounding *terms*. As such, the model allows for efficient queries and can be updated incrementally in negligible time to reflect changes to the underlying document collection. We discuss the versatility of this approach for event summarization, the completion of partial event information, and the extraction of descriptions for named entities and dates. We create and provide a LOAD graph for the documents in the English Wikipedia from named entities extracted by state-of-the-art NER tools. Based on an evaluation set of historic data that include summaries of diverse events, we evaluate the resulting graph. We find that the model not only allows for near real-time retrieval of information from the underlying document collection, but also provides a comprehensive framework for browsing and summarizing event data.

Keywords

Event extraction; event representation; document indexing; named entities; entity linking; summarization; ranking

1. INTRODUCTION

The description of events is one of the core concepts of human communication, as we desire to not only speak about the way things *are*, but also about how they *change*. It thus comes as no surprise that the detection, extraction and analysis of events plays a pivotal role in natural language processing. As a result, a lot of research has already been devoted to these tasks, and existing works focus on the extraction of events and temporal facts from more or less unstructured textual sources such as news articles [24, 27], social media [16], Twitter [3, 32], Wikipedia articles [13, 28, 41] or even Wikipedia edits [19, 25]. Some authors take the opposite approach and link Wikipedia events to historic news articles [29]. The breadth of these applications is reflected in the number of different definitions for the term *event* in the literature. Here, we view an event in analogy to the definition given for the TDT task [8] as *"something that happens at a given place and time between a group of actors."* Thus, we regard events as something at the intersection of *time*, *location* and involved *actors*, either individually or in *organizations* of multiple persons. In these cases, the void of structural information in unstructured textual sources is one of the greatest challenges, albeit one that can be ameliorated by focussing on the structure of the events themselves. On a language level, this is reflected in the original ACE definition of event detection [10] and has been utilized for tasks such as event threading [30] and incident threading [12]. On a higher level and across documents, however, with prior knowledge of already extracted named entities that are not necessarily contained within the sentences that correspond to the traditional definition of events, a local approach alone is not sufficient.

Given the definition of an event through the involved entities, the information pertaining to a specific event may be spread across several pages and not provided in one single context. Consider, for example, the documentation of the Olympic Games in Wikipedia. For each iteration of the games, there are pages that go into great detail about the games themselves, along with a multitude of pages about individual athletes. However, not all information about events at the games can be found on the central page, while the pages of individual athletes may be lacking important information such as exact dates or even the location, and merely reference to the Olympic Games of a given year. Thus, only the combination of data from multiple pages allows the reconstruction of all pieces of an event, including the place, time, and involved participants. As a result, the tasks of event extraction and summarization are closely tied to named entity extraction and entity linking. For large document collections, such a global approach thus requires the inclusion of efficient indexing strategies of the involved entities and documents in the process.

SIGIR '16, July 17-21, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2911451.2911529

When considering events as the co-occurrences of named entities of different types, we are also no longer bound by the concepts of *one sense per discourse* [15] or *one sense per collocation* [42]. While these have been shown to be valid in most situations and thus acceptable assumptions, they leave room for improvement as the task of disambiguation becomes less taxing when multiple entities of differing classes are involved in a given context. Given the close link between event detection and named entity extraction, a representation of the data that accounts for this seems sensible, especially if it enables queries to the data in a similar fashion. This approach is supported by recent research into web query evaluation, where the importance of the distinction between so called *content words* and *intent words* has been highlighted [34]. We find that this concept can be applied to the browsing of event data, where membership of query entities to a certain class is known at query formulation time. By using a structure for queries that reflects the structure of the event data itself, we can leverage this information that would otherwise be lost.

While numerous systems and approaches exist that enable browsing of extracted information or query results based on temporal characteristics as a timeline [3, 26, 35, 40], there are others that allow browsing on a spatial dimension instead [2]. With regard to events, however, we argue that the temporal, geographic and social components are inextricably interwoven and retain their full semantic meaning only within this entire context. Any method for the extraction and representation of events should therefore include all of these components (or *entities*) and assign to them equal importance in the underlying model. Here, we provide an important step in this direction and demonstrate the flexibility of such an approach, not just for event extraction but also for related tasks such as summarization and the generation of timelines.

Contributions. Our contributions are twofold. **(i)** We define the LOAD model for the representation and indexing of named entities for the task of event retrieval and description, which is versatile and well suited to related tasks such as event and entity summarization or entity linking. **(ii)** We provide an instantiation of our model on the English Wikipedia, which we make available to the research community[1], including the code for creating and using such a graph, as well as the evaluation data sets.

Structure. In Section 2, we discuss prior and related work, before we present our model in detail in Section 3. Afterwards, we discuss implementation strategies and an application of the model to the English Wikipedia in Section 4. In Section 5, we evaluate the LOAD graph for Wikipedia against a ground truth that is based on a *This Day in History* data set. We give a summary and outlook in Section 6.

2. RELATED WORK

Related work can be split into two broad categories, namely the extraction of events or temporal facts and graph-based information retrieval frameworks.

Extraction of temporal facts and events. Temporal information is prevalent in many documents across domains and provides a method of inducing structure in unstructured document collections due to the ordering aspects of time. Thus, exploiting temporal information extracted from documents has become an important part of information retrieval [6], which has recently seen a lot of advances in this regard. Therefore, the following list is focussed on works that are based on linking temporal and entity information for event retrieval and description.

Using a news archive, Setty et al. generate timelines highlighting important dates for a specific user query (e.g., about persons or events) [35]. Their approach exploits the document creation times of news articles under the assumption that the top-k time-travel query result for the topic of interest changes significantly at important times. Huet et al. use structured data in the form of a knowledge base to mine temporal trends or assess the importance of entities [22]. In an approach that is also based on an external knowledge base of news articles, Gupta and Berberich identify time intervals of interest for given keyword queries based on pseudo-relevant documents [21]. They employ a probabilistic approach for the selection of suitable documents for a given query and generate a time interval from the contained temporal expressions. Kanhabua and Nejdl analyse temporal anchor texts extracted from Wikipedia's edit history to track and detect the evolution of entities and events [25]. As in the approaches described above, temporal metadata (in this case, the edit history) is used to discover time-related knowledge.

In contrast to these approaches, our method uses only a document collection and exploits the temporal information that is available in the unstructured content of the documents. Articles that are similar in this respect include the analysis by Filannino and Nenadic, who extract temporal footprints of objects, persons, or historic periods from encyclopedic descriptions of Wikipedia articles [13]. Abujabal and Berberich also address the problem of extracting events from semantically annotated document collections. Based on methods from frequent itemset mining, they identify and rank events with relation to named entities [1]. In contrast to our approach, however, they consider events to be represented by sentences and thus cannot extract events that are spread across multiple documents. Kanhabua et al. assess the importance of temporal expressions to events based on entities and features [26]. They extract events based on the co-occurrence of entities and locations, but restrict co-occurrences to those within documents and single sentences.

While these approaches are similar to our work in exploiting temporal information described in the documents' text, we do not analyse single Wikipedia pages or use concept templates, but rather study the relationship between dates and content (e.g., entities, events, or keywords) in a general and global way. For this, we consider the full document collection at once without limiting our knowledge extraction to specific concepts. In addition, our model can be used for a multi-way summarization approach for points in time, entities and concepts.

There are also a number of systems for building knowledge bases of temporal information or timelines. One such system, CATE, extracts the context of entities (e.g., persons) and presents them in a timeline structure [40]. Kuzey and Weikum provide a framework for the extraction of temporal facts and events from the content and structure of Wikipedia articles by means of patterns and rules [28]. A similar approach is used to extend YAGO to include temporal knowledge [41]. Unlike these systems, we do not aim

[1]The Wikipedia LOAD graph, a Java implementation of our algorithm and interface, and the evaluation data can be found at `http://dbs.ifi.uni-heidelberg.de/?id=load`

to provide a knowledge base but a method of exploring and browsing a previously unknown document collection of unstructured text with respect to not only time but all entities that participate in real-world events.

Graph-based information retrieval. Das Sarma et al. build an *entity dynamic relation graph* to identify specific entities that participate in trending events, but do not consider locations [9]. Jatowt et al. use a time-term association graph in an estimation of the focus time of entire documents [23]. This graph is constructed from information that is extracted from an external corpus of news articles. Based on this association information, they identify discriminative time-term associations and employ these to estimate the focus time of given documents. Dutta and Weikum perform cross document co-reference resolution based on spectral graph clustering of mentions [11]. In a study of query methods, Bendersky and Croft use hypergraphs to model the occurrence of arbitrary classes of entities in queries [4]. However, this model is only applied to the queries themselves and not to the underlying data representation of the documents. The two works that are likely the most similar to our approach with respect to the model are based on graph representations of documents that facilitate Information Retrieval tasks. Blanco and Lioma explore a graph-based approach that models terms as nodes in a graph and weights their connections based on co-occurrence counts [5]. Based on this approach, they analyse the fitness of structural graph metrics for document ranking, but do not include named entities. Similarly, Rousseau and Vazirgiannis use an unweighted but directed graph to account for term order in text representation, the so called *graph-of-word* model [33]. In contrast to our work, they focus on the sentence level and do not include entities, which limits the possibilities of this model in event extraction. A third paper that uses a method similar to ours is from the field of citation analysis, where Chakraborty et al. suggest a system for finding communities in citation networks that uses a tripartite graph approach of authors, venues and papers [7], which is similar in semantics to our definition of actors, locations, and times.

The significant difference between our model and the above mentioned approaches is the proximity-based method of representing the co-occurrences of entities in the documents. This factor has been shown to play a major role in the quality of the retrieved information [39] and has recently been used by Geiß et al. to extract semantically meaningful networks of relations between entities of identical types from Wikipedia [17, 18]. In this paper, we combine this concept of text-based proximity with graph-based indexing of entities that has been employed for the modelling of terms and temporal expressions in a bipartite model [36]. We extend this concept to a multi-partite approach that includes all classes of entities that are of relevance to the task of event extraction. To this end, we provide an efficient representation of co-occurring entities that allows for both ad-hoc browsing of the entire data and real-time updates of the database in the case of changes to the underlying document collection, and take a step towards obtaining some of the functionality behind Google's Knowledge Graph for previously unseen document collections without an external knowledge base.

3. MODEL

With the main task being the identification and representation of events that emerge from the joint occurrence of named entities and temporal expressions, we base our model on the relationship between distinct classes of such entities that can be found in document collections. The most prominent class of involved entities are the *actors* in an event. Generally, these correspond to the named entity type of *persons*, but we use the more general term to also include non-person actors in possible fictional settings, to which the model is equally applicable. The underlying assumption is that actors are singular individuals. In contrast, we consider groups of people to be *organizations*, meaning that in this model, an organization may describe a company, a political party or even a rock band. To represent the geographic component, we include *locations*, which in the most general formulation are points or areas in space at which events take place. Finally, the temporal dimension is included by *dates*. Here, we only consider dates and not intervals, although a refinement of the model in such a way that it includes a set of temporally ordered, discrete dates as an interval is certainly possible. Since we are interested in modelling and ultimately extracting the relationship between these entities, a graph is a natural choice for the model as described in the following.

3.1 The LOAD graph model

Let P be a collection of documents (or *pages*) for which we define the model. Each document $p \in P$ consists of a number of sentences $s \in p$. With S we denote the set of all sentences in all documents, i.e., $S := \bigcup_{p \in P} \{s \in p\}$. Note that, while two sentences may have identical content, we consider them to be separate entities under this model. We define an ordering $\tau : S \to \mathbb{N}$ that maps a sentence to its index in the document that contains it. Assuming an underlying bag-of-words model, we can now treat a sentence as a collection of words, which we partition into several classes in accordance to the motivation given in Section 1. Note that we do not make a distinction between words and expressions that may consist of multiple words. For the purpose of this paper, we consider words to be units within a text that has already been tagged for named entities and dates. Specifically, we distinguish between the classes *locations* L, *organizations* O, *actors* A, and *dates* D, hence the acronym *LOAD*. Any word in a sentence that is neither of the aforementioned is considered to be a *term*. Thus, we define the set of terms T as

$$T := \bigcup_{s \in S} \{w \in s \mid w \notin L \cup O \cup A \cup D\}. \tag{1}$$

A sentence can thus be considered to be a multiset of entities $s \in (L \cup O \cup A \cup D \cup T)^*$. In the following, we refer to the set of actors, locations, organizations, dates and terms as *entities*. Based on these seven classes, we define their union $V := L \cup O \cup A \cup D \cup T \cup S \cup P$ along with a function $\sigma : V \to \{L, O, A, D, T, S, P\}$ that maps each element $v \in V$ to its original class $\sigma(v)$. Note that we do not consider ambiguities at this stage. Therefore, if a word has two meanings (e.g., Washington, which could be a person's name or a location), we consider these meanings to be represented by two distinct elements of V that belong to different classes.

To obtain an undirected graph representation $G = (V, E)$ of the co-occurrences of these entities, sentences and pages, we introduce a set of edges as defined in the following. Since G is undirected, we do not make a distinction in notation between the edges (v, w) and (w, v). To retain the seven

partitions of V, we disallow edges between nodes in the same set, i.e., G is a multi-partite graph, such that for all $e = (v, w) : e \in E \Rightarrow \sigma(v) \neq \sigma(w)$. Since there exists a surjection from S to P, we also exclude direct edges between P and all further sets, as they are not necessary. Thus, we obtain for all $(v, w) \in E : w \in P \Rightarrow v \in S$. A schematic representation of the resulting graph structure is shown in Figure 1.

To generate weights $\omega : E \to \mathbb{R}$ for the edges of the graph, we introduce the concept of *Instances* I of entities. An instance describes a distinct occurrence of an entity in a sentence (and thus a page) and there exist surjective mappings $m : I \to V$ from instances to entities and $m_s : I \to S$ from instances to sentences. For example, if an entity v occurs twice on a given page p, there exist instances i and j along with sentences $s_i, s_j \in p$ such that $m_s(i) = s_i$ and $m_s(j) = s_j$ as well as $m(i) = m(j) = v$. With I_v we denote the set of all instances of entity v, i.e., $I_v := \{i \in I \mid m(i) = v\}$. To generate edge weights based on these instances, we furthermore distinguish between two different types of relationships. The first type of relationship occurs between sentences, pages and entities and describes a type of inclusion. For the inclusion of entities v in sentences s, we simply use the number of occurrences as a weight, i.e. we set

$$\omega(v, s) := |\{i \in I \mid m(i) = v \wedge m_s(i) = s\}| \tag{2}$$

We define the weight between sentences and pages analogously, but note that the weight always equals 1 iff the page contains the sentence and 0 if it does not. The second type of relationship occurs between multiple entities and describes a co-occurrence within sentences and pages. To arrive at a definition of weights in this case, we first define the distance δ between two instances as

$$\delta(i, j) := |\tau(m_s(i)) - \tau(m_s(j))| \tag{3}$$

i.e., the distance equals the number of sentences between the instances, or 0 if they occur in the same sentence. If i and j do not occur on the same page, we set $\delta(i, j) := \infty$. Based on this distance, we define the weight of edges between two entities v and w as

$$\omega(v, w) := \sum_{\substack{i \in I_v \\ j \in I_w}} \exp(-\delta(i, j)) \tag{4}$$

i.e., we assign to all co-occurrences of instances of v and w a weight that diminishes exponentially with the distance between the two instances and derive the total edge weight as the sum of weights for all instances. Without loss of generality, we may treat edges with a weight of 0 as non-existing in the resulting graph to obtain a sparser and more efficient representation for the subsequent analysis. We thus define the binary adjacency matrix of G with dimension $|V|^2$ as $A_{vw} := 1$ iff $\omega(v, w) > 0$ and $A_{vw} := 0$ otherwise. We call the number of edges that are adjacent to a node v the degree of v and write $deg(v)$. Analogously, we define the class-specific degree $deg_x(v)$ of a node v as the number of edges from v to other nodes w for which $\sigma(w) = x$.

3.2 Hierarchical completeness conditions

For some of the entities that we consider in this context, there exist hierarchies that can be beneficial to include in the graph representation. One type of hierarchy is given by the granularity of dates, which includes years, months or days. To account for these differences in granularity, we consider dates to be sets of time points for which an inclusion hierarchy exists. That is, each day is included in a month and the same relation holds between months and years. As a result, we partition D into three subsets $D = D_y \cup D_m \cup D_d$. This hierarchy can be reflected in the graph by requiring that for all edges $(d, v) \in E$, if there exists a $d' \in D$ such that $d \subset d'$, then we also have $(d', v) \in E$ and we adjust the edge weight by setting $\omega(d', v) = \omega(d', v) + \omega(d, v)$, e.g., an edge between a day and another entity or sentence also induces an edge between this entity and both the month and year that include the day. An analogous hierarchical completeness condition can be formulated for locations if a geographic hierarchy of places is known.

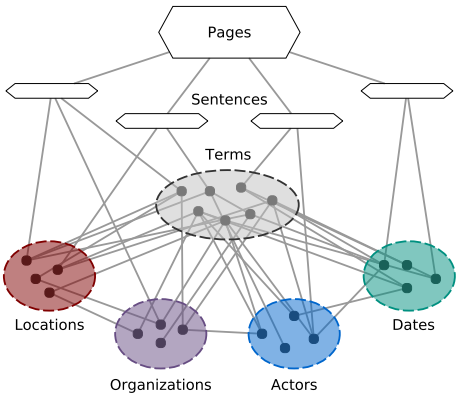

Figure 1: Schematic view of terms over the LOAD model along with their connections to the original sentences and documents.

A second kind of hierarchy can be defined for person names. Here, we observe that persons are frequently referred to by only parts of their full name, such as the first or last name. As an example, Alan Turing may be referred to as Alan Turing, Turing, etc. However, all mentions refer to the same person and induce co-occurrence relationships. Thus, we employ an approach that is reciprocal to the inclusion condition for dates and split a name n into components $n_1, ..., n_k$, for which we require that if there exists an edge $(n, v) \in E$ between the full name and some other entity v, then we also include edges $(n_1, v), ..., (n_k, v)$ and adjust the edge weight accordingly. While it is possible to apply this scheme to the names of organizations, these generally do not follow the same naming conventions as persons. However, an application to the names of locations is possible in cases where geographic hierarchy information is missing.

3.3 Ranking functions

To leverage the graph representation and extract information from the co-occurrences of entities in the document collection, we employ ranking functions. For any given entity, we can rank adjacent nodes in the graph representation by the importance of their co-occurrences to obtain the most relevant related entities. In other words, we obtain a ranking of nodes in one set, depending on the strength of their connections to nodes in other sets. The benefit of this approach is the possibility of computing relevant neighbours efficiently due to the graph representation, despite the overall size of the document collection.

First, we consider a binary ranking function between two sets of nodes. Thus, let $X, Y \in \{L, O, A, D, T\}$ be two

classes of entities such that $X \neq Y$. We define a binary ranking function as $r_{XY} : X \to \mathbb{R}^{|Y|}$, i.e., we map a node $v \in X$ to a vector of ranking scores such that each node in Y is assigned a relevance score with regard to v. Thus, we can determine the most relevant neighbours of a query entity v.

In order to instantiate the ranking function, we want to find a way of ranking relevant nodes efficiently on a local level, i.e., by using only the neighbourhood of that node. Therefore, we employ a graph adaptation of the *tf-idf* score that has been shown to work well in a bipartite graph setting for ranking temporal expressions and terms [36], which we further modify to include the distance-based weight between entities instead of simple co-occurrence counts. As a measure of the relevance of co-occurrences between two specific nodes, we build upon the analogy to terms that are contained in a document by equating nodes $y \in Y$ with documents that "contain" the nodes $x \in X$ they are connected to (i.e., the entities they co-occur with). We then observe that the weight $\omega(x, y)$ in the graph corresponds to the overall strength of co-occurrences between x and y in the document collection, meaning that it represents a kind of term frequency (tf). Similarly, the class-specific degree of a node (i.e., the number of adjacent edges that lead to nodes of the respective class) is equivalent to the frequency with which it appears alongside entities of that particular class. As such, it is a kind of document frequency and can be used to compute an inverse document frequency (idf). By combining the two, we arrive at a version of the *tf-idf* score that is adapted to the graph representation:

$$tf\text{-}idf(x, y) := \frac{1}{f}\omega(x, y) \log \frac{|Y|}{deg_Y(x)} \tag{5}$$

To obtain a normalized ranking in the interval $[0, 1]$, we divide the resulting scores by a normalization factor f, which is chosen as the maximum local *tf-idf* score:

$$f := \max_{y' \in Y}(tf\text{-}idf(x, y')) \tag{6}$$

By ranking all entities that are connected to a query entity according to their normalized *tf-idf* scores, we obtain the desired ranking. Since there is no difference between the partitions of the graph from a graph-theoretic point of view, we may input any of the classes as X and Y in any combination. We are, however, not limited to binary ranking functions. In the case of events that have a geographic and temporal aspect and pertain to actors and locations (or both), it may be required to rank the importance of entities in one set with regard to entities from multiple other sets. With the inclusion of terms, we thus need a function that is of higher arity, i.e., we require a function $r : X^n \to \mathbb{R}^{|Y|}$, where the X_i are pairwise disjoint with Y. Here, we can use a ranking that consists of a combination of the individual rankings as defined in Equation 5, i.e.

$$r(\vec{x}, y) := \frac{1}{n}\eta(\vec{x}, y) \sum_{i=1}^{n} r(x_i, y) \tag{7}$$

for a query vector $\vec{x} \in X^n$. The parameter η enforces a local cohesion, i.e., it ensures that only those candidate entities count towards the ranking that are connected to at least two query entities. Formally,

$$\eta(\vec{x}, y) := \begin{cases} 1 & \text{if } \sum_{i=1}^{n} \sum_{j>i}^{n} A_{yx_i} A_{yx_j} > 1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

In regard to summarization as an application, it is possible to extract a description of entities or a combination of entities as a ranking of adjacent terms with respect to the entity (or entities). However, it may be more convenient to directly extract entire sentences that are relevant to the provided entities instead of just keywords. Similarly, with respect to the task of entity linking, it may be pertinent to recommend well suited pages (i.e., documents) for a combination of entities. Thus, the ranking of sentences and documents is also of interest and can be achieved easily within the graph model. Formally, this requires a function $r : X^n \to \mathbb{R}^{|Y|}$, where the $X_i \in \{L, O, A, D, T\}$ and $Y \in \{S, P\}$. Here, we employ a straightforward instantiation of this ranking function as a proof-of-concept, although more involved measures are certainly possible within the model. To do this, we count the number of nodes in the query set that are connected to a sentence (i.e., entities that are contained in the sentence) or to all sentences within a page, respectively. For sentences s, we thus arrive at the following instantiation of a ranking:

$$r(\vec{x}, s) := \sum_{i=1}^{n} A_{sx_i} \tag{9}$$

For pages p, we can use the fact that the edges between sentences and pages are 0-1-valued to count the occurrences of all query terms in sentences of a page and obtain

$$r(\vec{x}, p) := \sum_{s \in S} \sum_{i=1}^{n} A_{sx_i} A_{sp} \tag{10}$$

Complexity. A key consideration for large document collections is the running time. The ranking functions as defined in Equations 5 and 7 can be computed efficiently in $\mathcal{O}(\langle deg_X Y \rangle \langle deg_Y X \rangle)$, where $\langle deg_X Y \rangle$ is the average set-specific degree of nodes in X with respect to set Y. For sparse graphs, these average degrees are smaller than the number of nodes by orders of magnitude, thus making these measures very efficient for the analysis of large document collections. Given the size of graphs that are obtained for larger document collections, this is a critical feature as we show in Section 4, where these ranking functions allow for ad-hoc ranking of neighbouring nodes. Beyond the presented ranking functions, the graph representation can serve as an index for further approaches based on entity co-occurrences.

4. APPLICATION AND EXPLORATION

In the following, we describe the technical details that are relevant to the implementation of our model. We then apply it to the English Wikipedia and discuss the resulting graph as well as query performance.

4.1 Document processing

Based on the model presented in the previous section, the implementation of a LOAD graph is fairly straightforward by extracting named entities from sentences and connecting them in a graph structure based on their distances in the text. However, we note at this point that the title of this paper was chosen due to the combinatorial explosion in the number of edges that result from fully connecting all entities that occur on a page, a process which creates a graph that is prohibitively large for large document collections. While the theory of the model is based on the assumption that entities on the same page share some connection, regardless of their distance in the text, long-distance connections are very weak

and mostly negligible due to the exponential decay in edge weights. Therefore, we include a cut-off parameter c in the algorithm that excludes edges if the distance between the two instances is too large. Similarly, terms are less likely to be related to entities outside of their own sentence, so we limit the edges between terms and entities to those that appear in the same sentence. Based on these considerations, we arrive at the LOAD algorithm for constructing a graph representation of a document collection (see Algorithm 1). In the following, we discuss a number of aspects of practical importance to the implementation.

Term extraction. The extraction of terms is not as straightforward as it initially appears. Since terms are defined as everything that is left over after entities have been removed from a sentence, the deletion of entities is key. Ideally, entities should not overlap, which would enable a structured deletion of entities from a sentence. In practice, overlaps happen, especially if several annotation tools are used. We find the simplest solution to be the use of a bitmap for characters in a sentence to mark the covered text of entities for deletion. Afterwards, marked parts of a sentence are removed and the remainder is considered for term generation. Stop words are removed to prevent high frequency noise from masking the co-occurrences of content words.

Stemming. To reduce the size of the graph and group related terms into one node for improved recall in query answering, we recommend the use of a stemmer for term processing. This makes it much easier to match semantically similar words in a query on the resulting graph. While lemmatization would be preferable during the document processing phase, it would not be compatible with queries on the graph, as lemmatization is hardly possible on query terms that are used outside of sentences.

Implementation architecture. As the resulting graph can be quite large, we consider two architectures. A purely memory-based approach is possible through the representation of the graph as a list of nodes with five types of adjacency lists, i.e., one adjacency list per adjacent class (thus, six lists in the case of nodes representing sentences and one for pages). Edge weights are stored in an identical structure. While such a representation is highly efficient with regard to query answering, it also requires large maps for the lookup of entity names, which tend to take up large amounts of memory. Alternatively, an external database can be used to store most of the information. With efficient indexing, the retrieval of node neighbourhoods is then slower but still possible. Hybrid approaches between the two architecture solutions are viable. For our experiments, we store only the sentences in an external database, as these are equivalent in size to the entire original document collection.

Algorithm 1 Creation of the LOAD graph based on the output of NER, temporal tagger and sentence splitter. The set function σ and distance function δ are defined in Section 3. The *update* function adds a new edge if it is not yet contained in E or adds the value to the existing weight ω if it already is. The cut-off parameter limits the number of co-occurrences.

Input: Documents D, cut-off parameter c
Initialize $V \leftarrow \emptyset$, $E \leftarrow \emptyset$ and $\omega(i,j) \leftarrow 0\ \forall i,j$
for $d \in D$ **do**
  $S_d \leftarrow$ sentences in d
  $N_d \leftarrow$ entities in d
  $V \leftarrow V \cup S_d \cup N_d \cup \{d\}$
  for $s \in S_d$ **do**
    update E with $(s, d, 1)$
    $N_s \leftarrow s \cap N_d$
    $T_s \leftarrow s \setminus N_s$
    $V \leftarrow V \cup T_s$
    for $t \in T_s$ **do**
      update E with $(t, s, 1)$
      for $n \in N_s$ **do**
        update E with $(n, t, 1)$
  for $n_1 \in N_d$ **do**
    $N_d \leftarrow N_d \setminus \{n_1\}$
    for $n_2 \in N_d$ **do**
      if $\sigma(n_1) \neq \sigma(n_2)$ and $\delta(n_1, n_2) \leq c$ **then**
        $w \leftarrow \exp(-\delta(n_1, n_2))$
        update E with (n_1, n_2, w)
Output: $G = (V, E, \omega)$

4.2 LOADing Wikipedia

To test the model on a large-scale document collection, we use the English Wikipedia dump of June 2, 2015, as input. Since the algorithm is designed to extract information from unstructured text, we exclude info boxes, tables, references, and pages of lists and thus only use the raw text without any links or pre-existing annotations. For the tokenization, sentence-splitting, POS-tagging, lemmatization and named entity recognition we use the Stanford Named Entity Recognizer [14]. We employ the 3-class model trained for CoNLL data to extract persons, organizations and locations per our definition of LOAD. As temporal tagger we use Heideltime [38] instead of StanfordNER, since Wikipedia articles largely follow a narrative structure and we therefore need a domain sensitive temporal tagger that distinguishes between the news and narrative domains [37]. For stemming terms, we use the Snowball stemmer that is an implementation of the Porter stemming algorithm [31]. We set the cut-off parameter for the distance between entities to $c = 5$, since this value allows the use of 32 bit single precision floating point numbers for storing the exponentially diminishing edge weights without loss of precision, which helps in keeping the graph size manageable. We use the hierarchical completeness condition for dates and the inclusion by splitting for the names of persons and locations.

In the annotation phase, we find that there are about 4.6M English Wikipedia articles that contain at least one annotation. The pages can be split into a total of 91.4M sentences, 53.5M of which contain at least one annotation. We find a total of 137.0M instances of entities, which are divided into 27.0M of class date, 44.2M of class location, 44.4M of class person and 21.3M that are annotated as organizations. After the extraction of co-occurrences and the aggregation of parallel edges, we obtain the LOAD graph, for which we give the metrics in Table 1. We observe that the number of distinct dates is by far the smallest due to the selected granularities year, month, and day. However, the coverage of dates in Wikipedia is above 50% for dates after the middle of the 16th century and perfect for dates after 1800 [36], meaning that each such date is mentioned at least once. The large number of terms can be explained by the presence of technical terms, mismatched names or loca-

class	L	O	A	D	T	S	P
L	0						
O	90.8	0					
A	275.8	105.7	0				
D	83.0	45.5	127.6	0			
T	182.8	93.9	316.6	57.3	0		
S	71.3	20.9	84.4	38.3	412.2	0	
P	0	0	0	0	0	53.5	0
nodes	2.7	3.4	7.1	0.2	4.9	53.5	4.5

Table 1: Number of edges (top) and nodes (bottom) of the LOAD graph constructed from the English Wikipedia (in millions).

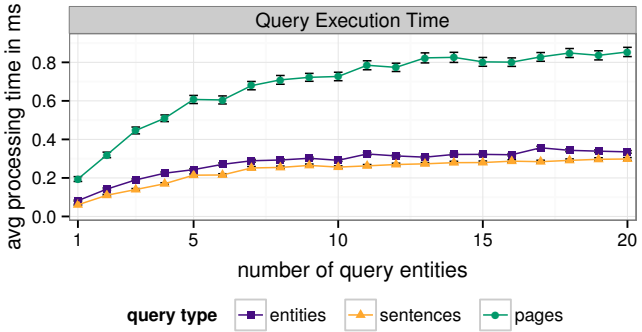

Figure 2: Processing speeds of the three query types with respect to the number of entities in the query. Results were averaged over 1000 samples, error bars denote the standard mean error.

tions and numeric data, which we did not model as separate entities (although it is a possible extension of the model).

The annotation of the data required approximately 4 days on a dual Intel Xeon E5-2650 CPU with 20 physical cores and 256GB main memory. The LOAD graph construction took 46 hours with a serial algorithm that is in theory trivially parallelizable for future approaches. The resulting graph has an uncompressed size of about 70GB on disk when edges are stored in one direction or 110GB if they are duplicated for faster initialization of the query interface. Despite these expensive initial computations, the resulting graph allows for near real-time information retrieval and browsing of relations as shown in Figure 2. Even for higher counts of entities in a query, the processing takes less than a millisecond on average (queries on multiple highly connected entities may take up to a second), due to the representation of the sparse graph in adjacency list format. Queries on page recommendations (i.e., for entity linking) take slightly longer since the connections between entities from all sentences on the page must be considered. For sentence queries, the shown time does not include the lookup of the actual sentences in the external database. In summary, we find that the processing time of information retrieval is insignificant given the size of the document collection and is thus competitive with search engine speeds, especially since queries are processed only serially in the current implementation.

4.3 Experimental results

Based on the LOAD graph of Wikipedia, we include a number of example applications. Due to the versatility of the system with regard to the classes of entities that can be used in a query, we only present a selection of possible application scenarios. However, we invite the reader to download the graph alongside our query interface and explore the data themselves or implement further ranking scores (see Section 1 for a link). For the following examples and the subsequent evaluation, we introduce the concept of a subquery. Here, this means that input strings for entities of classes *location* and *actor* in a query are split into their components which are then included in the query as well. The queries thus benefit from the completeness condition of the graph as defined in Section 3. In the following, we use the syntax $\langle OC : (IC, value)^* \rangle$ to describe queries, where $OC, IC \in \{L, O, A, D, T, S, P\}$ are the desired output class and the classes of input entities, respectively, while *value* is the name of the query entity.

Browsing. The most straightforward application of the graph is the ability to browse the connections between actors, locations, dates, and organizations. In Table 2, we show the top-ranked results of three queries centred on *Edward Snowden*. By including Snowden as the only query entity, we obtain a list of organizations that are closely tied to him, including the *NSA* and *USIS*, the company that vetted Snowden. Duplicate entries for the different spellings of the NSA are caused by the NER step, which suggests that the LOAD approach can be used as a tool for disambiguation or co-reference resolution. If we include *Barack Obama* as a second query entity, the focus of the results shifts from security agencies to politics, where the incident was discussed. Such browsing can thus be used as a tool in following connections and co-occurrences of entities through the data to explore events, much like a knowledge graph.

Summarization. Based on the relationships of entities and terms in the graph, extracting descriptions for entities is no different from asking for any other entity (see Table 2, right). However, we can also query for sentences that contain the relevant entities directly. For example, the resulting sentences for the query $\langle S : (A, \text{Edward Snowden}), (O, \text{NSA}) \rangle$ is *"In early 2013, thousands of thousands of classified documents were disclosed by NSA contractor Edward Snowden"*, which summarizes the relationship nicely. While the current approach of locating sentences that contain the specified entities is straightforward, more intricate summarization metrics can be adapted to the existing graph structure.

Entity and concept linking. Since we built the graph on Wikipedia, we can also use it to recommend Wikipedia pages for entities. The query $\langle P : (A, \text{Edward Snowden}) \rangle$ unsurprisingly yields Snowden's Wikipedia page as the highest ranked result. However, we can link concepts as well, for example with $\langle P : (A, \text{Edward Snowden}), (T, \text{Surveillance}) \rangle$. For this query, the Wikipedia page for *Global surveillance disclosures since 2013* is placed ahead of Snowden's page in the ranking, since it lists the surveillance activities that Snowden reported about in greater detail. In this context, the concept of subqueries is helpful, since persons are rarely referred to repeatedly by their full name on their own page. For example, the query $\langle P : (A, \text{Albert Einstein}) \rangle$ ranks the page for the *Albert Einstein Transfer Vehicle* highest, which was used to supply the International Space Station. The reason for this is that it is always referred to by the full name, while Albert Einstein himself is more often referred to as Einstein. Allowing subqueries fixes this and results in the physicist's page being top ranked. While the transport vehicle never should have been tagged as a person during NER, our findings indicate that the model is able to detect such problems.

query:	$\langle O : (A, \text{Edward Snowden}) \rangle$		$\langle O : (A, \text{Edward Snowden}), (A, \text{Barack Obama}) \rangle$		$\langle T : (A, \text{Edward Snowden}) \rangle$	
rank	organizations	score	organizations	score	terms	score
1	nsa	1.000	nsa	0.546	surveil	1.000
2	national security agency	0.288	senate	0.503	leak	0.985
3	gchq	0.182	congress	0.340	document	0.610
4	us national security agency	0.083	republican	0.290	whistleblow	0.532
5	usis	0.043	democratic party	0.283	contractor	0.496

Table 2: Top 5 highest ranked results for three queries centred on *Edward Snowden*. Weights are given as the normalized adapted *tf-idf* weights or their combination in queries with multiple entities. Terms are stemmed.

5. EVALUATION

In the following, we present two data sets of historic events and use them to evaluate the LOAD graph of Wikipedia.

5.1 Evaluation data

Since we constructed the LOAD graph on Wikipedia data, we cannot turn to the Wikipedia event pages that are frequently used for evaluation in this context. To our knowledge, there are no data sets that have already been annotated for entities and are suitable for an evaluation of Wikipedia content, so we annotated it ourselves. To avoid using Wikipedia data, we obtained a list of events from the World History Project, which provides trivia about events *On This Day in History* [20]. These websites contain lists of historic events for each day of the year as well as dates of birth and dates of death of more or less famous persons. The site has information about 6558 dates of birth with a short mention of the person's job and 1483 dates of death along with a sentence about that person's accomplishments or circumstances of death. There are 5805 diverse historic events from scientific discoveries to famous events. We describe the preparation and annotation in the following.

Historic Events. As historic events we view the set of dates that remain when dates of birth and dates of death are removed. For these events, we also have a date of granularity day and one sentence that describes the event. We randomly chose 500 such events and annotated them by hand for persons, locations, organizations, and remaining terms. An example is shown in Table 3. It is evident that this data is very diverse in both structure and content. Structurally, events consist of differing numbers of entities of different classes and have varying numbers of terms. Based on content, the data can be categorized into various areas of life and ranges from well known world events to minor descriptions, such as the patent on a bicycle crank. We specifically chose this set of random events instead of hand picked events to exclude further bias and provide a more challenging evaluation data set than the dates of death baseline.

Dates of death. The dates of death serve as a baseline since they are likely represented by a single sentence in Wikipedia. Since they end with the word *died*, we use pattern matching to remove them from the historic events. Afterwards, we manually annotate the data to strip titles from names, split the description from the name and separate multiple names and nicknames. For example, the sentence *"H(oward) P(hillips) Lovecraft, horror writer, died."* yields the information that Lovecraft was a horror writer, which we can use as terms to describe him, but it also provides the alias *H.P. Lovecraft* in addition to his full name. We store this information together with the provided dates for a total of 1483 persons.

data	method	found	miss	cor@1	prc@1
dates of death	LOAD	869	613	122	0.082
	LOADsq	1326	156	**207**	**0.140**
	LOADTsq	1374	108	125	0.084
	LOADTsq+	**1443**	**39**	13	0.009
	BASEw	869	613	206	0.140
	BASEr	869	613	63.25	0.043
historic events	LOAD	290	210	39	0.078
	LOADsq	341	159	**40**	**0.080**
	LOADTsq	**414**	**86**	33	0.066
	BASEw	290	210	19	0.038
	BASEr	290	210	0.48	0.001

Table 4: Evaluation results for both data sets. Shown are the used methods and the two baselines, the number of identified dates, the number of missing dates (not identified), the number of top ranked dates that were in the evaluation data sets (cor@1), and the resulting precision at rank 1 (prc@1).

5.2 Date prediction

To evaluate the performance of the LOAD model on Wikipedia based on the two event data sets, we use the description of events as query input and evaluate the resulting ranking of dates with respect to the known date. We chose this approach since there exists exactly one date of granularity day per event in the evaluation data set, and the coverage of dates in Wikipedia is very good for the considered time frame [36], which reduces the chance of evaluating the completeness of Wikipedia or the NER tool. For the ranking, we only consider days since this is the granularity in the evaluation data sets. We test several possible methods based on the LOAD approach. With *LOAD*, we denote the basic method of inputting all query entities and ranking dates according to the combined *tf-idf* ranking. For *LOADsq*, we allow subqueries. For *LOADTsq*, we use the descriptive terms of events in the query in addition to the entities. *LOADTsq+* is a special query in which we also include the terms *death* and *died*. Additionally, we use two baseline approaches. For *BASEr*, we rank dates that are connected to any query entity at random (this is iterated 1000 times and averaged). *BASEw* on the other hand ranks dates by their weighted connection ω to any entity in the query.

In Table 4, we show the results for both evaluation sets. Independent of the data set, we find that the LOAD methods are able to locate a higher number of correct dates and include it at some position in their ranking. The number rises as we include more entities and terms in the queries. In the case of death dates, the LOAD approach outperforms the randomized baseline but is strongly correlated with the weighted baseline. Since we only have one class of query entity in this case, this result is expected. Here, the *LOADsq* approach with subqueries performs best with regard to the

date	event	type	locs	orgs	actors	terms
1918-07-08	*Ernest Hemingway*, `Red Cross` volunteer, wounded in **Italy**.	war	1	1	1	2
1966-09-08	"Star Trek" debuted (`NBC`).	cul	0	1	0	3
1973-10-06	**Israel** attacked by **Egypt** and **Syria**.	war	3	0	0	1
1866-11-20	Bicycle with a rotary crank patented (*Pierre Lallemont*).	sci	0	0	1	4
1960-08-19	*Francis Gary Powers*, U2 spy plane pilot, convicted in a **Moscow** court.	pol	1	0	1	6

Table 3: Examples of events from the evaluation data set that are annotated for *actors*, locations and `organizations` together with the date on which the event took place. For each event, the number of entities by class is listed. Note that these events are not necessarily represented by a single sentence in Wikipedia.

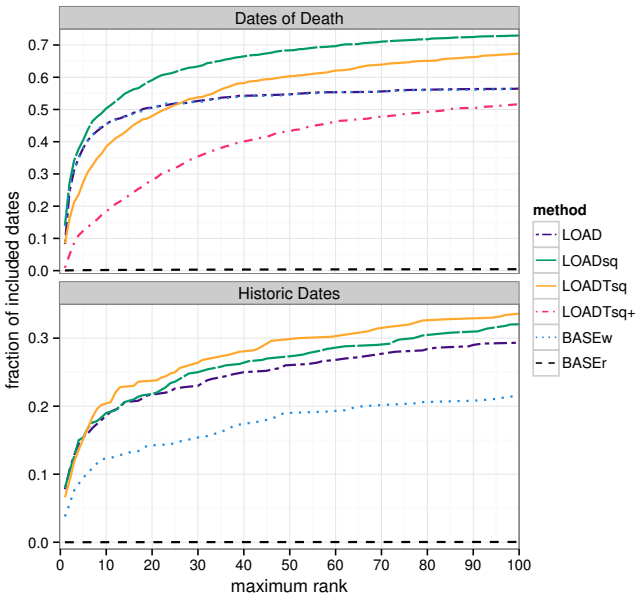

Figure 3: Prediction of dates based on event data. Shown is the number of identified correct dates in the top k ranks versus the rank k.

number of correct dates that are ranked at top position. The results are similar for the historical event data, although the weighted baseline performs much poorer with the addition of further query entities. Interestingly, the inclusion of terms in the query helps with the overall recall but not with the precision of the results. In order to evaluate the browsing potential of the method when used as a kind of search engine, we also consider the number of correctly identified dates with progressing rank in the list, as shown in Figure 3.

Here, we find that *LOADsq* performs best for the date of death data, for which it includes 50% of the correct dates in the first 10 positions of the ranking. The performance of *LOADTsq* is worse and even though it and *LOADTsq+* ultimately include the most correct dates, this happens at a point in the ranking where it is of little relevance, unless this were to be used as a pre-selection step for a further analysis. For either data set, the LOAD methods with subqueries perform significantly better than either baseline. For the historic event data, the inclusion of terms leads to a further improvement over the other LOAD methods, which emphasizes the need for terms in the model that may be important or even the only known features of an event.

6. CONCLUSION & ONGOING WORK

In this paper, we presented the graph-based LOAD model for the representation of named entities, dates and terms in the context of a document collection for the purpose of event extraction, browsing and indexing. We demonstrated the versatility of the system, pointed out further applications, and evaluated the system on two data sets of historic events.

A point that is worth discussing is the dependence of the LOAD model on the quality of the NER phase. While StanfordNER is a state-of-the-art system, certain topics on Wikipedia are simply too far removed from the training data to achieve good performance. For example, a query for entities related to the term *Klingon* yields results that will be familiar to any fan of Star Trek. However, the classes of such entities are assigned randomly, since Stanford is for obvious reasons unable to handle fictional Klingon names. This is unlikely to be an issue for smaller corpora of news articles, but for diverse document collections, further considerations are necessary. As such, we see a number of areas where the model can be improved in practical applications. Most prominent is the selection of terms, which can be reduced to a given set of parts-of-speech, based on the POS tags that are assigned in the NER phase. Doing so would allow for a more finely nuanced selection of terms. Different parts of speech such as verbs could then also be used to distinguish or describe certain kinds of events as composites of entities A second possibility is the inclusion of further classes of named entities, such as StanfordNER's *money* or *percent* classes. With regard to the model itself, the distinction between different classes of entities in the edge creation step (i.e., the multi-partiteness) is geared towards event extraction. For different applications, however, relationships between entities of the same class could be included. For example, a measure of similarity between persons might help to improve the quality of queries by accounting for synonyms and co-references.

We conclude with the observation that we tested the approach on Wikipedia, primarily because it is an enormous and free source of unstructured text that can be cleaned to remove noise. However, the LOAD model itself is useful for any document collection, since it makes no assumptions about document structure or origin. An application to a streaming or online setting is possible, where the model stands to benefit profoundly from the underlying graph representation, which allows incremental updates of any kind. The addition of new entire documents is possible in real-time, simply by processing the document and adding the resulting subgraph to the main graph structure. Even addition to or deletions from individual sections of a document can easily be accounted for, since the adjustment of edge weights happens locally within the graph structure. Thus, the model is able to handle the processing of frequently edited document collections like Wikipedia or news feeds.

Ongoing work. We are currently working on combining NER, temporal tagger, LOAD algorithm, query interface,

and word embedding for terms into a standalone version that can be applied to any document collection to facilitate efficient event browsing. Given the quality issues in the output of NER on Wikipedia, we are also working on the extraction of named entities from Wikipedia based on internal links, with the aim of constructing a more accurate version of the Wikipedia LOAD graph as a community resource.

7. REFERENCES

[1] A. Abujabal and K. Berberich. Important events in the past, present, and future. In *WWW*, 2015.

[2] B. Adams, G. McKenzie, and M. Gahegan. Frankenplace: Interactive thematic mapping for ad hoc exploratory search. In *WWW*, 2015.

[3] O. Alonso and K. Shiells. Timelines as summaries of popular scheduled events. In *TempWeb*, 2013.

[4] M. Bendersky and W. B. Croft. Modeling higher-order term dependencies in information retrieval using query hypergraphs. In *SIGIR*, 2012.

[5] R. Blanco and C. Lioma. Graph-based term weighting for information retrieval. *Information Retrieval*, 2012.

[6] R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. Survey of temporal information retrieval and related applications. *ACM Computing Surveys*, 2014.

[7] T. Chakraborty and A. Chakraborty. OverCite: Finding overlapping communities in citation network. In *ASONAM*, 2013.

[8] C. Cieri, S. Strassel, D. Graff, N. Martey, K. Rennert, and M. Liberman. Corpora for topic detection and tracking. In *Topic Detection and Tracking*. Springer, 2002.

[9] A. Das Sarma, A. Jain, and C. Yu. Dynamic relationship and event discovery. In *WSDM*, 2011.

[10] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *LREC*, 2004.

[11] S. Dutta and G. Weikum. Cross-document co-reference resolution using sample-based clustering with knowledge enrichment. *TACL*, 3:15–28, 2015.

[12] A. Feng and J. Allan. Finding and linking incidents in news. In *CIKM*, 2007.

[13] M. Filannino and G. Nenadic. Mining temporal footprints from Wikipedia. *COLING*, 2014.

[14] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*, 2005.

[15] W. A. Gale, K. W. Church, and D. Yarowsky. One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*, 1992.

[16] A. Gattani, D. S. Lamba, N. Garera, M. Tiwari, X. Chai, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Entity extraction, linking, classification, and tagging for social media: a Wikipedia-based approach. *PVLDB*, 6(11), 2013.

[17] J. Geiß, A. Spitz, and M. Gertz. Beyond friendships and followers: The Wikipedia social network. In *ASONAM*, 2015.

[18] J. Geiß, A. Spitz, J. Strötgen, and M. Gertz. The Wikipedia location network - overcoming borders and oceans. In *GIR*, 2015.

[19] M. Georgescu, N. Kanhabua, D. Krause, W. Nejdl, and S. Siersdorfer. Extracting event-related information from article updates in Wikipedia. In *Advances in Information Retrieval*. Springer, 2013.

[20] A. Guisepi. History world: On this day in history. `http://history-world.org/ontd.htm` (2015-10-02).

[21] D. Gupta and K. Berberich. Identifying time intervals of interest to queries. In *CIKM*, 2014.

[22] T. Huet, J. Biega, and F. M. Suchanek. Mining history with Le Monde. In *AKBC*, 2013.

[23] A. Jatowt, C.-M. Au Yeung, and K. Tanaka. Estimating document focus time. In *CIKM*, 2013.

[24] S. Julinda, C. Boden, and A. Akbik. Extracting a repository of events and event references from news clusters. *COLING*, 2014.

[25] N. Kanhabua and W. Nejdl. On the value of temporal anchor texts in Wikipedia. In *TAIA*, 2014.

[26] N. Kanhabua, S. Romano, and A. Stewart. Identifying relevant temporal expressions for real-world events. In *TAIA*, 2012.

[27] E. Kuzey, J. Vreeken, and G. Weikum. A fresh look on knowledge bases: Distilling named events from news. In *CIKM*, 2014.

[28] E. Kuzey and G. Weikum. Extraction of temporal facts and events from Wikipedia. In *TempWeb*, 2012.

[29] A. Mishra, D. Milchevski, and K. Berberich. Linking Wikipedia events to past news. In *TAIA*, 2014.

[30] R. Nallapati, A. Feng, F. Peng, and J. Allan. Event threading within news topics. In *CIKM*, 2004.

[31] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[32] A. Ritter, Mausam, O. Etzioni, and S. Clark. Open domain event extraction from Twitter. In *KDD*, 2012.

[33] F. Rousseau and M. Vazirgiannis. Graph-of-word and TW-IDF: new approach to ad hoc IR. In *CIKM*, 2013.

[34] R. S. Roy, R. Katare, N. Ganguly, S. Laxman, and M. Choudhury. Discovering and understanding word level user intent in web search queries. *Web Semantics*, 30:22–38, 2015.

[35] V. Setty, S. Bedathur, K. Berberich, and G. Weikum. InZeit: Efficiently identifying insightful time points. In *VLDB*, 2010.

[36] A. Spitz, J. Strötgen, T. Bögel, and M. Gertz. Terms in time and times in context: A graph-based term-time ranking model. In *TempWeb*, 2015.

[37] J. Strötgen and M. Gertz. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *LREC*, 2012.

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

[39] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *SIGIR*, 2007.

[40] T. A. Tuan, S. Elbassuoni, N. Preda, and G. Weikum. CATE: context-aware timeline for entity illustration. In *WWW*, 2011.

[41] Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. Timely YAGO: harvesting, querying, and visualizing temporal knowledge from Wikipedia. In *EDBT*, 2010.

[42] D. Yarowsky. One sense per collocation. In *HLT*, 1993.