A Fresh Look on Knowledge Bases: Distilling Named Events from News

Erdal Kuzey
Max Planck Institute for Informatics
Saarbrücken, Germany
ekuzey@mpi-inf.mpg.de

Jilles Vreeken
Max Planck Institute for Informatics and
Saarland University
jilles@mpi-inf.mpg.de

Gerhard Weikum
Max Planck Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Knowledge bases capture millions of entities such as people, companies or movies. However, their knowledge of named events like sports finals, political scandals, or natural disasters is fairly limited, as these are continuously emerging entities. This paper presents a method for extracting named events from news articles, reconciling them into canonicalized representation, and organizing them into fine-grained semantic classes to populate a knowledge base. Our method captures similarity measures among news articles in a multi-view attributed graph, considering textual contents, entity occurrences, and temporal ordering. For distilling canonicalized events from this raw data, we present a novel graph coarsening algorithm based on the information-theoretic principle of minimum description length. The quality of our method is experimentally demonstrated by extracting, organizing, and evaluating 25 000 events from a corpus of 300 000 heterogeneous news articles.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and Principles

Keywords

Knowledge Bases; Temporal Knowledge; Event Mining; Information Extraction; Minimum Description Length

1. INTRODUCTION

Motivation: Large knowledge bases such as `dbpedia.org`, `yago-knowledge.org`, or `freebase.com` (the core of the Google Knowledge Graph), are valuable assets for entity awareness in search, summarization, analytics, and recommendations. These assets contain millions of individual entities (people, places, products, etc.), including *named events* such as elections, revolutions, notable disasters, important concerts, etc.

However, the coverage of such events is fairly limited in today's knowledge bases, the reason being that entities and facts are mostly extracted from Wikipedia and similarly curated sources. None of the above projects taps into news or social media for increasing their population of named events. Thus, major events are captured only late, after being properly edited in Wikipedia, and events in the long tail, such as concerts of indie rock bands, and brand-new events, such as hurricanes or political scandals, are completely out of scope. We aim to capture more events of this kind as early possible.

Problem Statement: This paper addresses the problem of populating a knowledge base with fine-grained emerging events, along with detailed semantic typing (e.g., using classes like rock concerts or hurricanes), relationships among events (sub-events, temporal order, etc.), as well as people and organizations participating in events. The raw input for this task is articles from newspapers, magazines, and online feeds. A seemingly obvious approach would be to run a clustering algorithm on a set of news articles, using a text-based contents similarity measure. Each resulting cluster would then be interpreted as a separate event. However, our goal is to go beyond this and to obtain semantically clear output that is ready for populating a high-quality knowledge base. This entails the following desiderata:

- *Canonicalized event entities:* Rather than merely clustering text snippets or noun phrases from news articles, we aim to output event entities in canonicalized form: exactly all news referring to the same event are placed in the same equivalence class, and a representative name is chosen for the event.
- *Semantic labeling:* Each event is labeled with one or more semantic types (aka. lexical classes) from a knowledge base such as `freebase.org` or a high-quality thesaurus like WordNet. In this paper, we use the type system from `yago-knowledge.org` which integrates Wikipedia categories and WordNet classes. The type system forms a DAG ranging from broad classes such as `disaster`, `tournament`, or `performance` to fine-grained classes such as `volcanic_eruptions`, `football_finals`, or `benefit_rock_concerts`. In addition, events should be annotated with their participating entities like people, organizations, locations, etc. These in turn should also reside in the knowledge base, rather than being mere noun phrases.
- *Chronological ordering:* Events must be placed on positions or intervals on the time axis. This is a requirement for chronological ordering of related events (e.g., one causing the other), even if the news articles that constitute an event exhibit widely varying timestamps.

CIKM'14, November 3–7, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2661984.

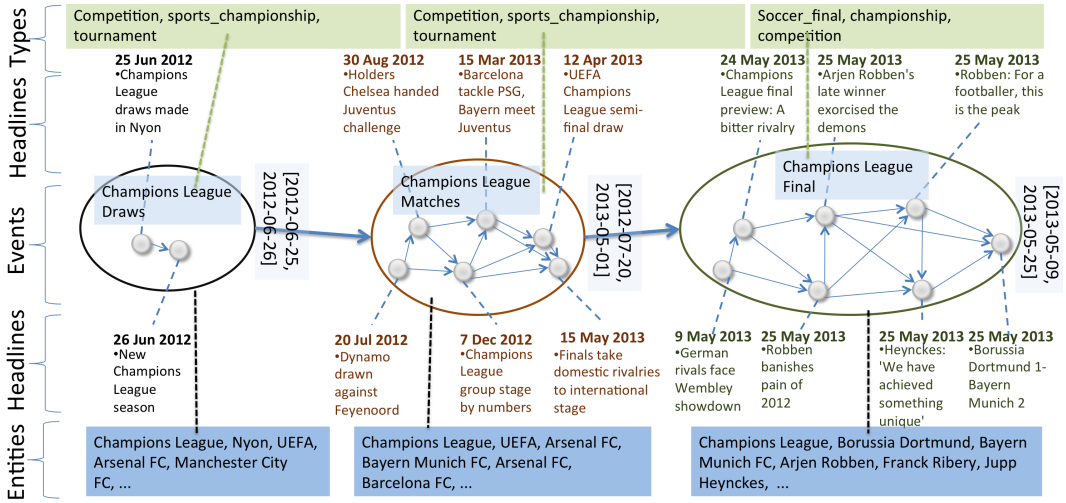

Figure 1: Output for the theme of "*UEFA Champions League 2012/2013*".

- *Event hierarchies:* Events should be organized in a hierarchical manner, with refinements into sub-events and inclusion in super-events. For example, Edward Snowden's request for asylum in several European countries is a sub-event of the Prism Scandal, and Steve Jobs' fight against cancer has sub-events like various therapies, his death, the memorial at Stanford, etc.

From a computational perspective, we address the following problem. Given a heterogeneous collection of news articles (from many different sources), *group* these into equivalence classes denoting the same events, *label* the equivalence classes with semantic types and participating entities, *chain* them in chronological order on the timeline, and *organize* them in a sub-/super-event hierarchy.

An example output that our method achieves, by processing several hundred articles about the UEFA Champions League season 2012/2013, is shown in Figure 1. The figure shows three different events and the news articles from which they were inferred, together with their representative names (inside ovals), their semantic types (top line), their participating entities (bottom line), and their chronological ordering. Note that the underlying articles in each cluster are not necessarily from the same date; so different clusters can heavily overlap in their overall timespans and inferring the temporal order between events is not obvious at all.

State of the Art: There are prior methods for mining events and storylines from a batch or stream of news articles (e.g., [3, 5, 18]). These methods compute output at the level of (ambiguous) noun phrases such as news headlines rather than true entities. For example, a typical output could be events such as "Bayern wins Champions League", "Bayern beats Borussia", "Bayern's triumph in London", "Bayern overcomes last year's final trauma", etc. These are semantically overlapping and do not denote distinct events. Simple clusters of news articles are not sufficient for a high-quality knowledge base.

In contrast to this prior work, our method reconciles all surface cues for the same event into a single entity in a canonicalized representation. Moreover, our method assigns fine-grained class labels such as *soccer finals* or *European sports championships* to an event. So there is a fundamental difference between traditional event mining and our goal of populating a knowledge base, satisfying the desiderata stated above.

Contribution: This paper presents novel methods for populating event classes (concerts, ceremonies, elections, conflicts, accidents, disasters, tournaments, etc.) of high-quality knowledge bases by extracting, cleaning, and canonicalizing fine-grained named events from news corpora. Our methodology involves the following contributions:

- mapping news articles into event classes by *automatically labeling* them with fine-grained types (Section 3);
- a multi-view graph model that captures relationships and relatedness measures between news articles, and a novel graph-coarsening algorithm for grouping and temporally ordering news articles based on the information-theoretic principle of minimum description length (Sections 4, 5);
- building a high-quality knowledge base with 25 000 named events automatically derived from 300 000 news articles referenced as external links in Wikipedia (Section 6).

2. SYSTEM OVERVIEW

Our system taps online news from a heterogeneous set of sources (newspapers, magazines, other news feeds). "Distilling" canonicalized and semantically organized events from these news articles proceeds in three steps. First, news are labeled with semantic types; these are later carried over to the distilled events. Second, news are grouped in a hierchical manner; each group will eventually be an event. Third, related events are chained in chronological order. The second and the third step are carried out by the same algorithm, based on coarsening a multi-view attributed graph (MVAG). Finally, the resulting events and their semantic annotations are placed in the knowledge base, populating a taxonomy of fine-grained semantic types.

Feature sets: We exploit four kinds of feature groups that are provided by each news article n or can be derived from it by information extraction methods:

- the *textual content* $\mathcal{C}(n)$ of the article,
- the *publication date* $t(n)$,
- the *set of entities* $\mathcal{A}(n)$: people, organizations, countries, companies, etc., appearing in the content, and

- the *semantic types* $\mathcal{T}(n)$ of events covered by an article, for example, bombing, earthquake, festival, concert, etc.

The entity names appearing in a news article are extracted using the Stanford NER tagger [7]. Since these are used as features for subsequent processing, we do not attempt to disambiguate entities onto canonical representations in a knowledge base, but simply accept a tolerable level of ambiguity and noise. Semantic types for news articles are obtained in two steps: first, by constructing statistical language models (LM's) [24] for articles and types; second, by mapping an article to its similar types using Kullback-Leibler divergence. This is further explained in Section 3.

Distance measures: Features are used to compute different kinds of distance measures between news articles.

The *content distance* is the cosine distance of the articles' $tf \cdot idf$ vectors over a bag-of-words model:

$$dist_{text} = cosine(\mathcal{V}(n_i), \mathcal{V}(n_j))$$

where $\mathcal{V}(n_i)$ and $\mathcal{V}(n_j)$ are the $tf \cdot idf$ vectors of news articles n_i and n_j, respectively; tf denotes the term frequency of words, and idf denotes the inverse document frequency of words (i.e., the inverse frequency of a word in the corpus).

The *temporal distance* is the normalized time distance between the publication dates of news articles:

$$dist_{temp}(n_i, n_j) = \frac{|t(n_i) - t(n_j)|}{H}$$

where H is the time horizon of the entire news corpus. We define H as the difference between the earliest and the latest timestamps appearing in the corpus.

The *attribute distance* between articles is the weighted Jaccard coefficient capturing the overlap in the *entity sets* or in the *type sets* respectively:

$$dist_{attr}(n_i, n_j) = \frac{\sum_{x \in \mathcal{X}(n_i) \cap \mathcal{X}(n_j)} weight(x)}{\sum_{x \in \mathcal{X}(n_i) \cup \mathcal{X}(n_j)} weight(x)}$$

where $\mathcal{X}$ can be entity set $\mathcal{A}$, or type set $\mathcal{T}$. Entity names are weighted by their $tf \cdot idf$ values, the types are weighted by their idf values.

These are all standard measures from the information-retrieval and text-mining literature. Our specific choices are based on prevalent usage in the state-of-the-art, but could be easily replaced.

Multi-view attributed graph (MVAG): The feature sets and the distance measures are used together to construct a multi-view attributed graph (MVAG) of news articles; $G = (V, A, E, F)$.

Vertices and attributes: A vertex $v_i \in V$ of the MVAG corresponds to a news article n_i. Each vertex inherits a set of attributes A from n_i: its textual content $\mathcal{C}$, its timestamp t, its associated entities $\mathcal{A}$, and its types $\mathcal{T}$.

Edges and weights: The MVAG has two kinds of edges: undirected ones, edge set E, and directed ones, edge set F. All edges are weighted. Two vertices are connected by an undirected edge $e_{i \leftrightarrow j}$ if they share at least one entity and at least one type:

$$\mathcal{A}(n_i) \cap \mathcal{A}(n_j) \neq \emptyset \wedge \mathcal{T}(n_i) \cap \mathcal{T}(n_j) \neq \emptyset$$

The weight of the edge is the content distance between two vertices; $w(e_{i \leftrightarrow j}) = dist_{text}(n_i, n_j)$.

Two vertices are connected by a directed edge $f_{i \rightarrow j}$ if their timestamps indicate that they are ordered on the timeline. The weight of a directed edge is the temporal distance between the time stamps of vertices: $w(f_{i \rightarrow j}) = dist_{temp}(n_i, n_j)$.

3. ASSIGNING SEMANTIC TYPE LABELS

We have devised a two-step method for mapping news articles onto semantic event types in a knowledge base. First, news articles are mapped to Wikipedia categories (ca. 32 000 event categories) using statistical language models (LM's) [24]. Second, Wikipedia categories are mapped to Wordnet event classes by the heuristic of Suchanek et al. [1]. WordNet provides ca. 6 800 classes under the type label "event" in its taxonomy, with 5-6 levels of subclasses. Examples are: `final`→ `match`→ `contest`→ `social_event`→ `event`, and `riot`→ `violence`→ `aggression`→ `action`→ `act`→ `event`. All WordNet classes are also integrated in YAGO [1, 11], the specific knowledge base we aim to populate.

3.1 Mapping News onto Wikipedia Categories

The first step is based on language models for news articles and categories. The LM of a news article captures i) the news content in terms of title and body words, and ii) all entities in the news article, including normalized date literals that appear in the article. LM's are a principled approach in IR [24], widely used for query-result ranking, cross-lingual retrieval, question answering, etc. An LM is a probability distribution over words or other text features. LM's are usually defined for documents and for queries, with probabilistic or information-theoretic distance measures between LM's. For our setting, we customize and extend the notion of LM's as follows.

Document models: The document model of a news article n is constructed over keywords and entities appearing in the article, so it is a probability distribution over $\{w : w \in n\} \cup \{e : e \in n\}$, where w denotes keywords, and e denotes entities.

For example, the news article *Borussia Dortmund 1-2 Bayern Munich* (`http://www.bbc.co.uk/sport/0/football/22540468`), starting with "Arjen Robben's late winner exorcised the demons that have haunted him and Bayern Munich in the Champions League as they won a pulsating all-Bundesliga encounter . . . ", has a document LM with:

- keywords: {winner, exorcised, demons, match, goal, . . . }
- entities: {Arjen_Robben, Bayern_Munich, Champions_League, Bundesliga, Wembley_Stadium . . . }

The LM's probability distribution for news article n is

$$P[s] = \mu P_W[s] + (1 - \mu) P_E[s]$$

where s is a word or entity, $P_W[s]$ and $P_E[s]$ are estimated probability distributions for words and entities in n, respectively, and μ is a hyper-parameter that controls the relative influence of each of the two aspects. The LM of a Wikipedia category is defined analogously and constructed from all Wikipedia articles that belong to the category.

Estimating LM parameters: The LM parameters for both news articles and categories are estimated from frequencies in the underlying texts, with Jelinek-Mercer smoothing (using the global frequencies in the entire collection of all news articles and all categories, respectively):

$$P_W[s] = \lambda P[s \mid news\ n] + (1 - \lambda) P[s \mid news\ corpus]$$

where λ is the smoothing coefficient. $P_E[s]$ is estimated analogously.

Comparing news and categories: To compare how close a Wikipedia category is to a news article, we use the Kullback-Leibler (KL) divergence (aka. relative entropy) between the

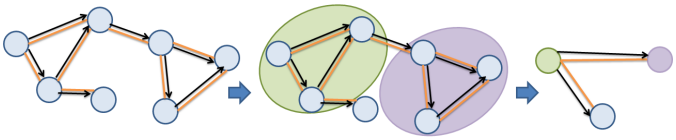

Figure 2: Coarsening a multi-view attributed graph.

corresponding LM's:

$$KL(news\ n\ ||\ category\ c) = \sum_s P[s \mid n] \cdot \log \frac{P[s \mid n]}{P[s \mid c]}$$

For each news article, we compute the top-k categories based on this distance measure, and accept those categories whose KL divergence is below a specified threshold.

3.2 Mapping Categories onto WordNet Classes

The second step of our two-step approach maps the accepted Wikipedia categories to their lowest (i.e., most specific) event type in the WordNet taxonomy DAG, by adopting and adjusting the heuristic of [1]. This method uses a natural-language noun-group parser on the category name to identify its head word, and maps the head word to the best matching WordNet type. If the head word is ambiguous, the word-sense frequency information of WordNet is used to make the choice. For example, for the category name "*General elections in Italy*", the word "*elections*" is found as the head word, and it is mapped to the first one of four possible senses in WordNet: election (a vote to select the winner of a position or political office) (see `http://wordnet.princeton.edu/`).

4. GROUPING NEWS INTO EVENTS

4.1 Design Alternatives and Choice

Once we have all the features of news articles in a collection, including the semantic type labels, our goal is to distill canonicalized named events from this collection and organize the events into semantic classes of the knowledge base. This task entails a *grouping* and a *chaining problem*: combining thematically highly related news into a single event, and ordering the resulting groups along the timeline.

A straightforward approach to this problem would be to proceed in two phases: first compute clusters of news articles based on similarity over all features, then infer ordering relations between clusters (e.g., by majority voting over the items in each pair of clusters). However, computing the ordering chains only after the clusters are determined may pose a poor if not unsolvable situation for the chaining step.

To overcome these issues, we designed a novel approach to this problem, by means of a graph coarsening. The rationale is that we transform the fine-grained MVAG for news articles into a coarser graph whose nodes correspond to the final event entities. Thus, our approach integrates the clustering and ordering tasks.

In order to identify the optimal coarsened graph for a given MVAG, we take a principled approach and employ Minimum Description Length (MDL) principle [9].

4.2 MVAG Coarsening

Given a multi-view attributed graph G with node set V, weighted undirected edges E, weighted directed edges F, entity sets $\mathcal{A}$, and types $\mathcal{T}$, a coarser graph G^* with $V^* \in 2^V$ (i.e., forming equivalence classes of nodes) and $E^*, F^*, \mathcal{A}^*, \mathcal{T}^*$ is computed such that i) G^* preserves the main properties and structure of G, and ii) G^* is simpler (smaller, coarser) than G.

The grouping of the original V nodes that leads to V^* induces the undirected edges E^*, directed edges F^*, edge weights, and attribute sets of the coarsened graph. This is explained in the following.

Consider the grouping function Γ for mapping V to V^*. Γ induces edge sets E^* and F^*:

$$e_{\Gamma(x)\leftrightarrow\Gamma(y)} \in E^* \Leftrightarrow \exists e_{x\leftrightarrow y} \in E$$

$$f_{\Gamma(x)\rightarrow\Gamma(y)} \in F^* \Leftrightarrow \exists f_{x\rightarrow y} \in F$$

Γ also determines the edge weights in G^* by averaging the weights of edges between all node pairs (x, y) that are mapped onto coarsened nodes $(\Gamma(x), \Gamma(y))$:

$$w(e_{x^*\leftrightarrow y^*}) = avg\{w(e_{x\leftrightarrow y}) | \Gamma(x) = x^*, \Gamma(y) = y^*\}$$

$$w(f_{x^*\rightarrow y^*}) = avg\{w(f_{x\rightarrow y}) | \Gamma(x) = x^*, \Gamma(y) = y^*\}$$

Γ induces entity sets $\mathcal{A}^*$ in G^*. It is worth noting that the entity set of a node in G can be noisy due to imperfect quality of the named entity recognition tool. In addition, there can be entities mentioned in a news article that are not relevant to the reported event. For example, the entities "*BBC_Radio, BBC_Sport_website*", extracted from the news article "*Champions League: Dortmund confident Mats Hummels will be fit*", are not relevant to the event mentioned in the article. Hence, the grouping function Γ induces the entity set of x^* as the shared entities of the fine nodes that are mapped to x^*. Therefore, the main participants (entities) of an event are captured, whereas irrelevant entities are avoided. Formally, $\mathcal{A}(x^*) = \bigcap \mathcal{A}(x_i)$, where $\Gamma(x_i) = x^*$. Γ induces types $\mathcal{T}^*$ in G^* in the same way as entity sets, using the intersection of type sets.

Figure 2 illustrates the MVAG coarsening.

4.3 Optimization Model

We formalize our problem using the Minimum Description Length (MDL) principle [9], which can be paraphrased as *Induction by Compression*. In a nutshell, it identifies the best model M^* in a family $\mathcal{M}$ of models M as the model minimizing $L(M) + L(D \mid M)$, where $L(M)$ is the length, in bits, of the description of M, and $L(D \mid M)$ the length of the data given M. This scheme ensures that M^* neither overfits nor is redundant—otherwise another M would minimize the sum. We formally define our problem as follows.

Problem: In our setting, the family $\mathcal{M}$ of models is the family of MVAG's. For a given graph G our goal is to compute the MDL optimal coarsened graph G^*

$$L(G, G^*) = L(G^*) + L(G \mid G^*)$$

is minimal. Hence, our algorithms aim to find a minimum of this objective function.

To use MDL we have to define how we encode a model, i.e., a coarsened graph G^*, and how we encode the input graph G given G^*. For the latter the high-level idea is to encode the error of G^* with respect to G, i.e., we encode their exclusive OR, $G^* \oplus G$, such that the receiver can reconstruct the original graph without loss upon receiving G^* and $G^* \oplus G$. We formalize these encodings as follows.

4.3.1 Encoding the coarse graph

To encode a coarse graph G^*, we encode all its properties: its vertices V, their attributes A, the undirected edges E,

and the directed edges F,

$$L(G^*) = L(V^*) + L(A^*) + L(E^*) + L(F^*)$$

The vertices. Encoding the vertices entails encoding their number (upper bounded by $|V|$), and encoding per vertex $v \in V^*$ to how many and which nodes in V it maps. Hence,

$$L(V^*) = \log(|V|) + \log\binom{|V|-1}{|V^*|-1} + \sum_{v \in V^*} \log\binom{|V|}{|v|}$$

The attributes. We encode the vertex attributes by

$$L(A^*) = \sum_{v \in V^*} \left(\log(|A^*|) + \log\binom{|A|}{|\mathtt{atr}(v)|} + \sum_{a \in \mathtt{atr}(v)} \log(\mathtt{rs}) \right)$$

where per coarse vertex $v \in V^*$ we encode how many attributes it has, which these are, and their weights. We encode weights at the resolution of the data, $\mathtt{rs} = 10^{\#sign.\ digits}$, based on the number of significant digits of the data.

The undirected edges. For encoding the undirected edges we first encode their number—using the upper bound $ub = |V^*|(|V^*| - 1)$—then identify which edges exist, and then encode their weights again using resolution $\mathtt{rs}$. We have

$$L(E^*) = \log(ub(E^*)) + \log\binom{ub(E^*)}{|E^*|} + |E^*| \log(\mathtt{rs})$$

The directed edges. Encoding the (one-)directional edges is almost identical; in addition we only need to transmit their direction, for which we require one bit per edge. Thus,

$$L(F^*) = \log(ub(F^*)) + \log\binom{ub(F^*)}{|F^*|} + |F^*| \log(\mathtt{rs}) + |F^*|$$

4.3.2 Encoding the data

To reconstruct the original graph, we need to transmit all information needed to interpret the coarse graph, as well as correct any errors it makes with regard to the input data. That is, we need to correct all missing and superfluous edges, attributes, and their weights. At a high level we have

$$L(G \mid G^*) = L(|V|) + L(A \mid A^*) + L(E \mid E^*) + L(F \mid F^*)$$

As $L(|V|)$ is constant for all models we can safely ignore it.

Attributes: Starting with the node attributes, we transmit the error per node by encoding i) the number of missing and superfluous attributes, ii) which attributes these are, and iii) the correct weights. Thus, $L(A|A^*) =$

$$\sum_{v \in V} \left(\log(|A \setminus A^*|) + \log\binom{|A \setminus A^*|}{|\mathtt{atr}(v) \setminus A^*|} + \sum_{a \in \mathtt{atr}(v)} \log(\mathtt{rs}) \right)$$

where $\mathtt{atr}(v)$ is the set of attributes of v. We encode the attribute weight errors using $\log(\mathtt{rs})$ bits each—if one is willing to make assumptions on the error distribution other choices are warranted.

Undirected edges: To reconstruct adjacency matrix E, we transmit the edges in the (upper diagonal part) of the error matrix $E_c = E^* \oplus E$ where $\oplus$ the exclusive OR. Thus,

$$L(E \mid E^*) = \log(ub(E)) + \log\binom{ub(E)}{|E_c|} + |E| \log(\mathtt{rs})$$

We encode the weight errors using $\log(\mathtt{rs})$ bits per edge in E.

Directed edges: Last, let us write $F_c = F^* \oplus F$ for the error matrix for the (uni-)directed edges. As above, we define $L(F \mid F^*)$ analogue to $L(E \mid E^*)$, but in addition need to specify the direction of edges in F which are unmodelled by F^* using one bit per edge. Hence, we have

$$L(F|F^*) = \log(|F \setminus F^*|) + \log(ub(F)) + \log\binom{ub(F)}{|F_c|} + |F| \log(\mathtt{rs})$$

Algorithm 1 GREEDY(MVAG G)

1: $Q \leftarrow \varnothing$
2: **for all** matched pairs $(u, v) \in V$ with $gain(u, v) > 0$ **do**
3: $Q.insert((u, v), gain(u, v))$
4: **while** $Q \neq \varnothing$ **do** ▷ Iterative coarsening phase
5: $mergeSet \leftarrow Q.popHead()$
6: **for all** overlapping pairs $(n, m) \in Q$ with $mergeSet$ **do**
7: $mergeSet = mergeSet \cup \{n, m\}$, remove (n, m) from Q
8: MERGE(G, $mergeSet$), UPDATE(G)
9: $recompute(Q)$
return G

In sum, we now have a principled and **parameter-free** objective function for scoring the quality of coarse graphs for a multiview attributed graph.

5. GRAPH COARSENING ALGORITHMS

For an input graph G, the set of all possible models $\mathcal{M}$ is huge. Moreover, it does not exhibit any particular structure (e.g., sub-modularity) that we can exploit for efficient pruning of the search space. Hence, we resort to heuristics. The algorithms proceed in iterations; each iteration aims to reduce the input MVAG by the following operations:

- MATCH: For a given node, the match operation finds a suitable node(s) to merge into a coarser node. The MDL based objective function matches the nodes that result in the largest gain.
- MERGE: This operation merges two or more nodes into a coarser node.
- UPDATE: Following a MERGE step we update the MVAG to reflect the new edges, edge weights, attributes, and attribute weights that result from node merging.

Within this general framework, we developed two specific algorithms: a greedy method and a randomized method, which we explain next in turn.

5.1 The GREEDY Method

A standard greedy method would have a MATCH operation that, for a given node, always chooses the node(s) for which a MERGE results in the largest gain of the objective function. While our method follows this general principle, it has a specific twist by performing a light-weight form of look-ahead on the options for subsequent iterations. Specifically, we determine in each iteration if the currently best merge can be combined with other merges whose node pairs overlap with the node pair of the best merge. Without this look-ahead, the algorithm produces many tiny event groups.

The algorithm keeps track of the candidate node pairs considered for merging, using a priority queue Q of node pairs sorted in descending order of gain (line 2-3 in Algorithm 1). The gain is calculated as the objective function's improvement caused by a potential node pair merge. The algorithm first selects the pair at the head of the queue, which decreases the objective function the most. Next, in contrast to standard greedy techniques, our algorithm scans the queue for other pairs that have one overlapping node with the nodes of the head pair (line 6-7). If such a pair is found, it is added to "*mergeSet*". The algorithm then proceeds further and repeats considering further merges, until it exhausts a bounded part of the queue.

Algorithm 2 RANDOMIZED(MVAG G, α, ϵ, T)

1: $best \leftarrow \varnothing$
2: **while** $T > \epsilon$ **do**
3: pick a random node u, and its match v
4: **if** $gain(u, v) > 0$ **then**
5: MERGE(G, $\{u, v\}$), UPDATE(G), $best \leftarrow G$
6: **else if** $Random.probe < e^{\frac{gain(u,v)}{T}}$ **then**
7: MERGE(G, $\{u, v\}$), UPDATE(G)
8: $T \leftarrow T * \alpha$
return $best$

As an example, suppose the algorithm found $<n_1, n_2>$ as "*bestPair*", and the next positions in the priority queue are $\{<n_2, n_5>, <n_4, n_6>, <n_3, n_1>\}$. We scan Q and identify n_5 and n_3 for merging as their best matches are already included in "*mergeSet*". The algorithm thus expands the "*mergeSet*" into the set $<n_1, n_2, n_3, n_5>$ (line 7) and merges all these nodes in one step (line 8). The subsequent UPDATE operation incrementally computes the necessary changes of the MVAG and updates all data structures for the graph. The gain values for nodes in Q are recomputed for the next level of coarsening. When there is no further coarsening operation that would improve the objective function, the resulting MVAG is returned.

5.2 The RANDOMIZED Method

Our randomized method is based on the principle of simulated annealing [13]. This is an established framework for stochastic optimization, traversing the search space (of possible MERGE operations) in a randomized manner. In contrast to greedy methods, the method can accept, with a certain probability, choices that lead to worsening the objective function. The probability of such choices is gradually reduced, so the algorithm is guaranteed to converge.

Our algorithm, in each iteration, performs a randomized coarsening step. It does so by picking a random node u and then identifying its best MATCH v for a MERGE operation (line 3 in Algorithm 2). This is slightly different from standard methods where both nodes of a node pair would be chosen uniformly at random. Our experiments have shown that our choice leads to faster convergence. If the considered coarsening step decreases the objective function, it is accepted and the energy of the system is updated (line 5). Otherwise, it is accepted with a probability of $e^{\frac{gain(u,v)}{T}}$.

After each accepted coarsening step, the UPDATE operator adjusts the graph. The algorithm maintains a so-called "temperature value" T (a standard notion in simulated annealing) which is gradually reduced after each iteration, by geometric progression with factor α. Note that T is initially chosen very high and α is chosen very close to 1, therefore, the temperature cools down very slowly after each iteration. The algorithm terminates when the temperature drops below a specified threshold ϵ. Across all iterations, the best solution is remembered, and this is the final output when the algorithm terminates.

6. EXPERIMENT SET 1: KNOWLEDGE BASE COVERAGE

Our overriding goal in this work is to populate a high quality knowledge base with high coverage of named events extracted from news. Thus, we performed a large-scale computation in order to populate a knowledge base with named events. We used the GREEDY method to process 295 000 news articles that are crawled from the external links of Wikipedia pages. These news articles are from a highly heterogeneous set of newspapers and other online news providers (e.g., `http://aljazeera.net/`, `http://www.independent.co.uk`, `http://www.irishtimes.com`, etc.). The Wikipedia pages themselves were not used.

# of events	24 348
# of sub-events	3 926
# of follow-up events	9 067
avg # of entities per event	18
# of distinct classes	453

Table 1: Statistics for extracted events.

Ranking event candidates: The set of extracted named events exhibit mixed quality and are not quite what a near-human-quality knowledge base would require. However, it is easy to rank the event candidates based on the following scoring model, and use thresholding for high precision.

We use the *cumulative gain* for a coarse node representing an event in the output graph as a measure of quality. This is defined as the total improvement of the objective function after the merge operations that involved intermediate nodes that finally belong to the coarse node at hand. Thus, the score for event $\mathcal{E}$ is $S(\mathcal{E}) = \sum_{q_i \in \mathcal{E}} gain(G, q_i)$, where G is the graph, q_i is an intermediate node created by a merge operation that contributes to final output node $\mathcal{E}$.

The acceptance threshold for an event score is set to the 90th percentile of the score distribution for all extracted events. This is a conservative choice, motivated by the reported quality of 90 to 95% for knowledge bases like YAGO [11]. Note that our task here is substantially harder than that of the YAGO extractors, as the latter operate on Wikipedia infoboxes. Thresholding may break chaining information between some events. Thus, chaining is re-computed by transitively connecting missing links. Each event is labeled by a "*representative news headline*". The representative news article is chosen based on the centrality score in the original MVAG.

Statistics: Our method computed 24 348 high-quality events from the underlying 295 000 news articles. A good portion of these events contain sub-events and follow-up events (i.e., events that appear on a chain). Table 1 shows other statistics about the extracted events. The most densely populated classes of events are: protest (8732), controversy (7514), conflict (4718), musical (1846), sport event (766), war (727), etc. An event can be assigned to multiple classes.

Assessing the KB coverage: We show that the events distilled by our methods yield much higher coverage than traditional knowledge bases which solely tap into semistructured elements of Wikipedia (infoboxes, category names, titles, lists). We compared two knowledge bases on their in-

	YAGO [2012-1-1,2013-9-4]	**Event_KB** [2012-1-1,2013-9-4]
total #	624	6423
% *year* events	16%	48%
% *month* events	10%	36%

Table 2: Coverage of events in YAGO vs. Event_KB.

Q: French military intervention in the Northern Mali conflict.			
Event_KB: France army in key Mali withdrawal			
News	**Entities**	**Classes**	**Time**
• India pledges $1 million to UN mission to Mali • Gunfire breaks out as Tuareg rebels enter city • France army in key Mali withdrawal ...	French Army, Mali, Tuareg, UN	conflict	[2013-01-11, 2013-05-25]
Q: Meteor explodes over the Russian city of Chelyabinsk.			
Event_KB: Russian Asteroid Strike			
News	**Entities**	**Classes**	**Time**
• Russian Asteroid Strike • Exploding Meteorite Injures A Thousand People in Russia • UN reviewing asteroid impact threat ...	Russia, Chelyabinsk	death, disaster	[2013-02-15, 2013-02-17]

Table 3: Two sample Event_KB events for given queries. YAGO has no results for these queries.

cluded events: YAGO [11], built from the 2013-09-04 dump of Wiki-pedia, vs. Event_KB, the knowledge base compiled by our method as described above.

For comparing the two knowledge bases in their coverage of events, we used the *events* section of Wikipedia's *year* articles as a form of ground truth. Specifically we used the pages `en.wikipedia.org/wiki/2012` and `en.wikipedia.org /wiki/2013`, but excluded events after 2013-09-04, the date of the Wikipedia dump for YAGO. In total, these articles contain 51 itemized snippets, each of which briefly describes a salient event. We also considered the monthly pages, e.g., `en.wikipedia.org/wiki/January_2013`, for the total relevant time period, together providing 9811 events in the same textual format.

Note that these text items are not canonicalized events: they mention an event but often do not point to an explicit Wikipedia article on the event, and in many cases such an explicit article does not exist at all. For example, the Wikipedia text for January 11, 2013 says: "The French military begins a five-month intervention into the Northern Mali conflict, targeting the militant Islamist Ansar Dine group." However, the only href links to other Wikipedia articles (i.e., entity markup) here are about the French army, the Ansar Dine group, and the Northern Mali conflict in general. The event of the French intervention itself does not have a Wikipedia article.

For each of these textually described events in the *year* pages, we manually inspected YAGO and Event_KB as to whether they cover the event as an explicit entity or not. For the 9811 event descriptions in the *month* pages, we inspected a random sample of 50.

Results: The coverage figures are shown in Table 2. For both ground-truth sets (*year* and *month* pages of Wikipedia), Event_KB shows more than 3 times higher coverage. This demonstrates the ability of our approach to populate a high-quality knowledge base with emerging and long-tail events.

The total number of events that Event_KB acquired for the relevant time period is 10 times higher than what YAGO could extract from semistructured elements in Wikipedia. Table 3 shows two sample Event_KB events, along with their semantic annotations, entities, and time spans.

7. EXPERIMENT SET 2: LABELING, GROUPING, CHAINING

7.1 Setup

In this section, we present experiments to evaluate the output quality of the various components of our methodology, in comparison with different baselines. We do this systematically by looking at our three main components separately: labeling, grouping, and chaining.

Test Data. We prepared two test datasets: news articles from i) *Wikinews* and ii) news sources referenced in Wikipedia articles. Note that Wikinews (`en.wikinews.org`) is totally independent of Wikipedia. Moreover, we removed all semi-structured elements from Wikinews articles to create a plain text corpus. The Wikipedia-referenced news are a highly heterogeneous set, from a variety of newspapers and other online news providers. We, therefore, refer to this dataset as *WildNews*.

For the Wikinews dataset, we picked articles by starting from topic categories that denote named events. Such categories are identified by matching years in their titles, e.g., *FIFA World Cup 2006, War in South Ossetia (2008), G8 Summit 2005, etc.* In total, we extracted around 70 named event categories from Wikinews, with a total of 800 articles. Some named events are simple such as *2010 Papal UK tour* or *Stanley Cup Playoffs 2007*, whereas others are complex and contain sub-events, e.g., *2010 Haiti earthquake* or *US War on Terror*.

For the WildNews dataset, we start with the collection of Wikipedia articles listed in the *Wikipedia Current Events* portal (`http://en.wikipedia.org /wiki/Portal:Current_events`). All news items cited by these Wikipedia articles with external links are crawled and together constitute the news corpus. This corpus has 800 news for 26 named events.

Ground Truth. The way we derived the news articles from named event categories already provides us with ground truth regarding the grouping and chaining of articles. However, the data does not come with semantic labels for populating a knowledge base. To this end, we have randomly chosen a subset of 3 news articles per named event, a total of 210 articles for the Wikinews data and a total of 78 articles for the WildNews data. These samples were manually labeled with one or more semantic classes from WordNet taxonomy (which is integrated in the YAGO knowledge base).

For example, the news article *Bomb blasts kill several in Iran* is labeled with `WN_bombing, WN_death, WN_conflict`.

Methods under Comparison.

1. Labeling. Our methodology that uses statistical language models to find semantic labels of news articles is compared with the $tf \cdot idf$ based cosine similarity.

2. Grouping. We compare our Greedy and Randomized methods (see Section 5) against several baselines: k-means clustering, hierarchical clustering, METIS [12] and the Storyline detection [21] method. All features (text, time, entities, types) are fed into the following flat distance measure that is used for the baselines:
$d(n_i, n_j) = \alpha \cdot d_{text}(n_i, n_j) \ + \ \beta \cdot d_{time}(n_i, n_j) \ +$
$\gamma \cdot d_{ent}(n_i, n_j) \ + \ \theta \cdot d_{type}(n_i, n_j)$

The reported results in the experiments are obtained by uniform weighting. We varied these parameters to study their sensitivity. This led to small improvements (up to 5% in precision) in some cases and small losses in other cases. Hence, we only report results for the uniform setting.

a. k-means clustering is run over both datasets with different k values. For each k value, the algorithm is run 10 times with different random centroid initializations. k-means achieved the best results, when k is set to the real number of the named events in the datasets. Thus, we set $k = 70$ for the Wikinews, $k = 26$ for the WildNews dataset.

b. Hierarchical agglomerative clustering (HAC), unlike flat clustering algorithms, can find clusters with different levels of granularity. We tried different linkage criteria: single-link (SLINK), complete link (CLINK), unweighted pair group method average (UPGMA), and weighted pair group method average (WPGMA). WPGMA achieved the best results for our settings.

c. METIS [12] is a graph partitioning tool that first coarsens a graph and then applies partitioning algorithms to it. METIS takes k number of partitions as input. We incrementally changed k to see how clustering performance change. METIS can have the best result, similar to k-means, when k is close to the number of real clusters. Thus, we set $k = 70$ for the Wikinews, $k = 26$ for the WildNews dataset.

d. Storyline detection [21] is a method to find story chains in a set of news articles returned by a query. Representing news articles as nodes of a graph, the method applies two steps to compute chains: It first finds the minimum set of dominating nodes of the graph. Second, it defines the dominating node with the earliest time stamp as a root. Other dominating nodes are combined by a directed Steiner tree algorithm. The edge weights in the multiview graph are induced via the flat distance measure mentioned above. Note that Storyline detection is designed to return one story chain at a time. We modified the method to apply it repeatedly for each dominating node. Thus, it can find multiple chains in the test data without any querying.

3. Chaining. We compare the methods that can find dependencies between news articles. Among the models we introduced so far, Greedy, Randomized, and Storyline detection can find temporal dependencies. Thus, these are the only models used for chaining comparisons.

7.2 Quality Measures

7.2.1 Labeling

As the labels computed by $tf \cdot idf$ cosine similarity and our LM-based method are ranked, we use the notion of *precision@k* to compare the k top-ranked labels against the ground-truth set of labels: $precision@k = (\sum_{j=1 \to k} r_j)/k$ where j is the position, and $r_j = 1$, if the result at the j^{th} position is correct and $r_j = 0$ otherwise.

We define *recall@k* analogously: $recall@k = (\sum_{j=1 \to k}^{k} r_j)/n$ where j is the position, and n is the number of ground-truth labels. Although n is usually close to k, it can be smaller than k for certain news articles that have only few true labels. In this case, one may suspect that the models can easily reach 100% recall. However, in practice, none of the methods compared in this paper was able to reach 100% recall for the top-5 labels.

7.2.2 Grouping

To compare different methods for grouping news articles into events, we look at pairs of news articles. We have ground-truth statements stating which articles belong to the same group (i.e., named event) and which ones are in different groups. Thus, we can compare the output of different kinds of grouping algorithms against the ground truth. We define precision and recall as follows.

Precision: the estimated probability of pairs (n_i, n_j) of news articles being in the same ground-truth event given that they are assigned to the same group.

Recall: the estimated probability of pairs (n_i, n_j) of news articles being assigned to the same group given that they come from the same ground-truth event.

Let G_M and G_T denote the groups for method M or ground truth T, respectively. Then we compute (micro-averaged) precision and recall as:

$$precision = \frac{\sum_{G_M} |\{(n_i,n_j) \in G_M \mid \exists G_T : (n_i,n_j) \in G_T\}|}{\sum_{G_M} |\{(n_i,n_j) \in G_M\}|}$$

$$recall = \frac{\sum_{G_T} |\{(n_i,n_j) \in G_T \mid \exists G_M : (n_i,n_j) \in G_M\}|}{\sum_{G_T} |\{(n_i,n_j) \in G_T\}|}$$

7.2.3 Chaining

For evaluating the quality of ordering events on the timeline, we again consider pairs (n_i, n_j) of news articles, the ordering of their corresponding groups by method M and the ordering of their true events in the ground-truth data T. We refer to these orderings as C_M (chaining by M) and C_T (chaining in T). So $(n_i, n_j) \in C_T$ means that there are ground-truth groups G_T, G'_T such that $n_i \in G_T$, $n_j \in G'_T$ and G_T is connected to G'_T by a directed "happened-before" edge. We compute precision and recall analogous to grouping. Instead of using G_M and G_T groups for method M or ground truth T, we use C_M (chaining by M) and C_T (chaining in T) in the formulas.

For all three tasks – labeling, grouping, chaining – our very primary goal is high precision as we aim to populate a high-quality knowledge base. Nonetheless, recall is also important for high coverage of events. The combined quality is usually measured by the harmonic mean of precision and recall, the F1 score: $F1 = (2 * precision * recall)/(precision + recall)$.

7.3 Results

7.3.1 Labeling

We compared our method to the baseline of using $tf \cdot idf$ based cosine similarity. Both methods mapped the 78 Wild-

	Wikinews						WildNews					
	LM			Cosine			LM			Cosine		
k	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
1	.71	.28	**.40**	.56	.21	.31	.59	.16	**.26**	.32	.09	.14
3	.58	.62	**.60**	.47	.50	.49	.54	.45	**.49**	.28	.22	.25
5	.58	.72	**.64**	.44	.62	.51	.47	.61	**.53**	.24	.30	.27

Table 4: Precision, recall, and F1 scores at k, for labeling.

News and 210 Wikinews news articles, for which we had ground-truth labels, to Wordnet semantic classes through Wikipedia event categories (Section 3). The top-5 semantic classes of each of the two methods are compared against the ground-truth classes. Table 4 shows precision and recall for different ranking depth k.

We see that our method substantially outperforms the cosine-similarity baseline.

7.3.2 Grouping

We ran flat clustering (k-means), multi-level clustering (METIS), hierarchical clustering (HAC), Storyline, Greedy, and Randomized on the Wikinews and the WildNews datasets. The final clusters produced by the methods are automatically evaluated by comparing them to the ground truth as explained in Section 7.2.

Although we gave k-means and METIS the "oracle" benefit of choosing k to be exactly the number of true events in the ground truth, they achieve inferior precision for both datasets. k-means obtained 62% precision, METIS 59%, and HAC 58% for the Wikinews dataset. For the WildNews dataset, k-means obtained 68% precision, METIS 71%, and HAC 50%. As precision is the crucial property for knowledge base population, k-means, METIS, and HAC are not suitable methods to populate knowledge bases for our setting.

The results for the remaining methods are shown in Table 5. Storyline prioritizes precision at the expense of poor recall. Although Storyline achieves a good precision for the WildNews dataset, it loses on F1 score due to low recall. Our methods, especially Greedy, yield similarly high precision at much higher recall. This results in roughly doubling the F1 scores of Storyline.

	Wikinews			WildNews		
	prec.	recall	F1	prec.	recall	F1
Storyline	.79	.10	.18	**.91**	.15	.26
Greedy	**.80**	**.31**	**.45**	**.91**	**.38**	**.54**
Randomized	.79	.29	.42	.77	.26	.39

Table 5: Precision, recall, and F1 scores for grouping.

	Wikinews			WildNews		
	prec.	recall	F1	prec.	recall	F1
Storyline	.94	.32	.47	.96	.29	.45
Greedy	**.96**	**.77**	**.85**	**.97**	**.67**	**.79**
Randomized	.93	.71	.81	.94	.44	.60

Table 6: Precision, recall, and F1 scores for chaining.

7.3.3 Chaining

We compare the methods that can find ordering dependencies between news articles. This rules out METIS, k-means, and HAC, and leaves us with the Storyline detection method as our competitor. As Table 6 shows, all methods achieve similar precision for chaining experiments for both datasets. However, Greedy and Randomized methods attain much higher F1 scores than Storyline.

7.3.4 Discussion

We have three main observations on our experiments:

1. The baselines METIS, k-means and HAC group news articles in an overly aggressive manner, resulting in clusters that unduly mix different events. Thus, they yield poor precision values. The Storyline method, on the other hand, favors pure clusters and is conservative in its chaining. This leads to low recall. In contrast, Greedy and Randomized methods exploit the rich features encoded in MVAG model well and jointly infer groups and chains, which results in high precision values and the best F1 scores for both datasets.

Considering that Randomized requires simulated annealing parameters as input, and those parameters may vary based on the news corpus, Greedy is the most practical parameter-free method with very good overall grouping and chaining performance.

2. All methods except Randomized perform well on grouping for the WildNews dataset, which seems surprising. The reason is that WildNews articles are longer than Wikinews articles and contain more entities. The average number of entities per article is 19 for Wikinews and 26 for WildNews. This observation suggests that the semantic features like entities in articles boost the grouping performance.

3. All methods perform better on the Wikinews dataset on chaining. The reason is that WildNews articles span a much shorter time period than Wikinews articles, and many articles have overlapping time stamps, which degrades the chaining performance. This observation implies that chaining is more difficult when news articles have nearby or overlapping timestamps.

The data of our experimental studies is available on `http://www.mpi-inf.mpg.de/yago-naga/evin/`.

8. RELATED WORK

Ontological event extraction: Knowledge bases such as YAGO [1, 11], DBpedia [4], or `freebase.com` contain entities of type event, fine-grained classes to which they belong, and facts about them. However, the coverage is fairly limited. The rule-based method by [14] has recently shown how to increase the number of named events and temporal facts that can be extracted from Wikipedia infoboxes, lists, article titles, and category names. However, all these approaches can capture a new event only after it has a sufficently rich Wikipedia article (with infobox, categories, etc.).

Story mining from news: There is a large amount of work on topic detection and story mining on news, e.g., [5, 18, 21, 23, 15, 6]. However, the events found by these systems are not ontological events, for which we require canonical representations and semantic typing, clean enough for populating a knowledge base.
Graphs for events: Identifying events in news streams has been addressed by modeling news articles as graphs of entities [3] or as graphs of keywords [2]. Both methods identify dense clusters in graphs. However, an event here is merely a set of temporally co-occurring entities and/or keywords. Thus, events are implicitly represented; they are not canonicalized and cannot be used to populate a knowledge base.
General graph algorithms: Attributed graphs have been used by [25, 19], for purposes unrelated to our topic. Graph coarsening has been pursued by [12, 17] for plain graphs in the context of graph-cut algorithms. Our setting is different by being based on multi-view attributed graphs (MVAG's). We developed novel methods for coarsening with a loss function specifically designed for our problem of grouping and chaining the nodes of an MVAG.
Graph summarization: Graphs can be summarized in terms of motif distributions such as frequent triangles or frequent subgraphs, or by showing aggregated views of a graph. The latter is related to our coarsening method. [20] presents the Snap-k operator for summarizing an attributed graph in terms of k groups. The key differences to our setting are that this work considers only graphs with one kind of edges (as opposed to our notion of MVAG's) and that the user determines the number of groups. [16] addresses the task of lossy graph compression by means of summarization using the MDL principle. In contrast to our setting, this work is limited to only plain graphs.
Mapping documents to Wikipedia categories: There are numerous studies on using Wikipedia categories for clustering documents, e.g., [10, 22, 8]. These methods exploit the semantic relatedness of documents to Wikipedia concepts and the link structure of Wikipedia categories. We use statistical language models to map news articles to YAGO classes, via specific Wikipedia categories for events.

9. CONCLUSION

The methods presented here fill an important gap in the scope and freshness of knowledge bases. We tap into (the latest) news articles for information about events, and distill the extracted cues into informative events along with temporal ordering. Our experiments demonstrated that our methods yield high quality, compared to a variety of baseline alternatives, and can indeed populate specific event classes in a knowledge base with substantial added value. Use-cases of this contribution include strengthening the entity-aware functionality of search engines, and also using the additionally acquired knowledge of events for smarter recommendations, e.g., when users browse social media, and for better summarization. Finally, the events distilled by our methods are themselves entities serving as background knowledge for better acquiring more events and other knowledge from news.

Acknowledgements

Jilles Vreeken is supported by the Cluster of Excellence "Multimodal Computing and Interaction" within the Excellence Initiative of the German Federal Government.

10. REFERENCES

[1] F. M. Suchanek, et al. Yago: A Core of Semantic Knowledge. *WWW*, 2007.
[2] M. K. Agarwal, et al. Real Time Discovery of Dense Clusters in Highly Dynamic Graphs: Identifying Real World Events in Highly Dynamic Environments. *PVLDB*, 2012.
[3] A. Angel, et al. Dense Subgraph Maintenance under Streaming Edge Weight Updates for Real-time Story Identification. *PVLDB*, 2012.
[4] S. Auer, et al. DBpedia: A Nucleus for a Web of Open Data. *ISWC/ASWC*, 2007.
[5] A. Das Sarma, et al. Dynamic Relationship and Event Discovery. *WSDM*, 2011.
[6] Q. Do, et al. Joint Inference for Event Timeline Construction. *EMNLP-CoNLL*, 2012.
[7] J. R. Finkel, et al. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *ACL*, 2005.
[8] E. Gabrilovich, et al. Overcoming the Brittleness Bottleneck Using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. *AAAI*, 2006.
[9] P. D. Grünwald. The Minimum Description Length Principle. *MIT Press*, 2007.
[10] X. Hu, et al. Exploiting Wikipedia as External Knowledge for Document Clustering. *SIGKDD*, 2009.
[11] J. Hoffart, et al. Yago2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artificial Intelligence, Vol. 194, p:28–61*, 2013.
[12] G. Karypis, et al. Multilevel Graph Partitioning Schemes. *ICPP, Vol. 3, p:113–122*, 1995.
[13] S. Kirkpatrick, et al. Optimization by Simulated Annealing. *Science, Vol. 220(4598), p:671–680*, 1983.
[14] E. Kuzey, et al. Extraction of Temporal Facts and Events from Wikipedia. *TempWeb Workshop*, 2012.
[15] W. Lu, et al. Automatic Event Extraction with Structured Preference Modeling. *ACL*, 2012.
[16] S. Navlakha, et al. Graph Summarization with Bounded Error. *SIGMOD*, 2008.
[17] I. Safro, et al. Advanced Coarsening Schemes for Graph Partitioning. *SEA*, 2012.
[18] D. Shahaf, et al. Connecting the Dots Between News Articles. *KDD*, 2010.
[19] A. Silva, et al. Mining Attribute-Structure Correlated Patterns in Large Attributed Graphs. *PVLDB*, 2012.
[20] Y. Tian, et al. Efficient Aggregation for Graph Summarization. *SIGMOD*, 2008.
[21] D. Wang, et al. Generating Pictorial Storylines Via Minimum-Weight Connected Dominating Set Approximation in Multi-View Graphs. *AAAI*, 2012.
[22] P. Wang, et al. Using Wikipedia Knowledge to Improve Text Classification. *KAIS, Vol. 19 (3), p:265–281*, 2009.
[23] R. Yan, et al. Evolutionary Timeline Summarization: A Balanced Optimization Framework via Iterative Substitution. *SIGIR*, 2011.
[24] C. Zhai. Statistical Language Models for Information Retrieval. *Morgan & Claypool Publishers*, 2008.
[25] Y. Zhou, et al. Graph Clustering Based on Structural/Attribute Similarities. *PVLDB*, 2009.