Learning Event Templates on News Articles

Mitja Trampuš, Dunja Mladenić
Department of Knowledge Technologies
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
E-mail: {mitja.trampus, dunja.mladenic}@ijs.si

ABSTRACT

We propose a pipeline for learning event templates from a large corpus of textual news articles. An event template is a machine-usable semantic data structure, in our case a graph, describing a certain event type. For instance, most earthquake news reports mention something in direction of "x people dead" or "town y shook at time z". Such templates can be used as an input for information extraction tasks or automated ontology extension. We present preliminary results of applying the proposed pipeline on a subset of News articles.

1 INTRODUCTION

Given the large amount of information encoded in written English and present on the web and elsewhere, there is a clear and long-understood need for machines to canonicalize that information as autonomously as possible in order to be able to use its inherent value.

One of the main approaches toward this end is (high-level) *information extraction*, where an algorithm is developed to fill a structured template (e.g. a database table row or a small ontology subgraph) with information extracted from unstructured text. The templates and the corresponding learning examples (tagged text), however, have to be prepared manually. In this work, we propose a step towards learning (automatically identifying) such templates prominent in a collection of news articles. Newswire is a particularly suitable domain for this task because many articles get written about each separate event, enabling us to exploit redundancy when determining the importance of pieces of information.

2 RELATED WORK

Automatic construction of templates for information extraction is already relatively well-researched (e.g. [6, 8]). However, the goal of existing approaches is to obtain **syntactic** templates for detecting words or phrases of a certain type (e.g. book titles). Our goal is to construct **semantic** templates (in the form of graphs) describing whole events; the templates do not act on the raw article text, but rather on semantic graphs describing separate events. We also aim to obtain templates that are useful in themselves, for ontology extension, not only information extraction. Furthermore, we learn the templates in a completely unsupervised manner as opposed to existing weakly supervised approaches.

Graph-based templates are also used in [7] in a context similar to ours, though the semantics are shallower. Also, the authors focus on information extraction and do not attempt to generalize the templates. Identification of templates in textual product descriptions is addressed in [10] in form of identifying product attributes and their values.

3 OVERVIEW

We propose an approach based on a pipeline for constructing abovementioned event templates in the form of small semantic graphs. Nodes represent actors or objects (nouns) and the links between them represent actions (verbs); see Figure 3 for an example of an automatically constructed template. Additionally, each node is rich with statistics about the context within separate articles it appears in, which will in future hopefully be a good starting point for training information extraction methods.

To test the proposed approach, we have used the Google News portal (although any news aggregation service would do). At this stage, we have limited ourselves to processing 7132 news articles from all topical categories, mostly published in March 2009.

4 THE PIPELINE

Each of the pipeline phases is described through an illustrative example. Consider the subset of articles reporting on various bombing attacks: in the next subsections, we will follow the information they convey and the form this information takes as it passes through the pipeline.

To avoid confusion, let us first detail some terminology: an *article* is a single web page which is assumed to report on a single *story*. A story is an event that is covered by one or more articles. Each story may fit some *event template*.

For example, the *event template* describing bombings in general may be supported by a *story* of a suicide bomber[1] in Baghdad and a *story* of NATO bombing Kabul. The story on Baghdad is in turn covered by a hundred or so web *articles* which are no longer an abstract concept but chunks of HTML code. Schematic overview of the pipeline is in Figure 1.

4.1 Data acquisition and preprocessing

We first need to obtain the data; to that end, we crawl *http://news.google.com* approximately every 40 minutes to obtain links to articles as well as a grouping of articles into stories. Each article is then downloaded from the publisher's website and cleaned of all HTML markup, advertisements, navigation and similar. We have developed a heuristic algorithm for **identifying the content part** of most any news article; the basic idea is to traverse the DOM tree and extract

[1] We apologize in advance for such a morbid example; sadly, it is exactly topics of this kind that get terrific news coverage and are therefore both familiar to everyone and convenient to analyze.

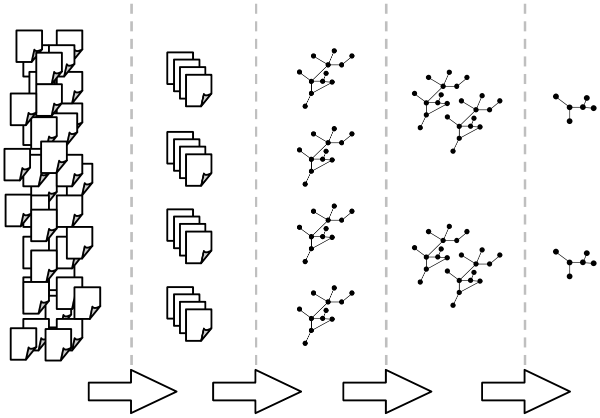

Figure 1. ***The five main stages of the pipeline.*** *Cleaned articles (1) are grouped (2) according to the story they cover. A semantic graph is constructed for each story (3). Topically related story graphs are clustered (4); the largest subgraph common to most of the graphs in each cluster (5) is the event template.*

the first block-level element (TD or DIV) containing a lot of text and very little of anything else, particularly links and images. This approach successfully identifies the title and the body of an article with accuracy of about 90%.

In the end, some additional cleanup is performed like encoding, whitespace and punctuation normalization.

4.2 Data annotation

Next, we enrich the text with semantic annotations of several types as follows. Using the ANNIE tool from the GATE [1] framework, we **detect named entities** and tag them as person, location or organization. Following that, we use the Stanford parser [2] to **extract triplets** (subject-predicate-object); the authors report the precision and recall of this stage to be about 85%. As a last step, we use the web service by Rusu [3] to perform **coreference and pronoun resolutions** ("Mr. Obama", "President Barack Obama" and "he" might all refer to the same entity within an article).

4.3 Story graph construction

Starting from a group of annotated articles on a single story, we want to construct a semantic graph relaying the gist of that story. This is similar to the classic problem of multi-document summarization; however, we have stronger assumptions about inter-document coherence (assumed to be high as all documents report on the same story) and we want to present the summary in the form of a semantic graph.

First we have to **identify the stories**, i.e. clusters of articles with high topical and temporal similarity. As already mentioned, we currently simply use existing Google's clustering results. Once a story has been identified, we once more perform coreference resolution on all of its articles simultaneously (since all mentions of e.g. Obama might have gotten mapped to "Mr. President" in one article and to "Barack Obama" in another).

We now have to **identify the important triplets**. Since each story is typically represented by at least 20 articles, typically 50-200, we can rely relatively heavily on statistics: the important triplets are those that appear many times throughout the articles. However, care must be exercised: in their attempt to meet the deadlines, journalists often copy-paste whole paragraphs from another source. Unfortunately, such plagiarism cannot be detected by string matching in its simplest form because short fragments of copied paragraphs often do get altered. Writers sometimes even creatively merge paragraphs from two or more sources. In any case, much of the text is repeated verbatim which would cause triplets from those passages to be rated too high. To mitigate the problem, we **compute paragraph similarities** based on character 4-gram overlap and weight paragraphs with $1/d_{sim}$ where d_{sim} is the number of paragraphs "very similar" to current one. The method, while simple, gives results with accuracy on par with what humans can do in such a loosely defined problem.

At this point, for the purposes of the algorithm, we discard the full article text and only keep the (weighted) triplets. The weight of a triplet is defined to be the sum of weights of all paragraphs it appears in, multiplied by "position score" (triplets that appear at the beginning of an article get a higher position score). Further, triplets with verbs like "report", "tell" suggest they are the result of sentences of the form "eyewitnesses told the police that …" and therefore uninformative; their overall weight is decreased drastically.

Triplet scores are further improved by making pairs of **similar triplets** increase each other's score. Similar triplets are identified using WordNet; the actual similarity score between two triplets is a product of experimentally set factors. The factors describe the number of words in which triplets overlap, the type of overlap (exact string match or via WordNet) and the position of overlap (e.g., it turns out that matching objects are more indicative of similar triplets than matching subjects). As WordNet does not provide uniform coverage of all topics, we have to compensate for that: triplets that appear similar to an extraordinary high number of other triplets are reduced in weight as its numerous similarities are most likely due to (too) rich synsets in that portion of WordNet. We also tried adjusting the similarity score in reverse proportion with the a priori probability of overlapping words, but that seemed not to affect performance noticeably (although evaluation was only informal). We do, however, employ a list of stopwords.

Finally, the scored triplets are viewed as tiny graphs; each graph has two weighted nodes (the scored subject and object) with a directed, weighted, labeled edge connecting them (label being the verb). Nodes are consolidated wherever possible, effectively creating a single connected component from most of two-node graphs.

We refer to the result as a *story graph*; an example can be seen in Figure 2. The central node in that graph is the subject “suicide bomb”, involved in several triplets including “target camp” (the top right heavily linked node), “killed people”, “blow mosque”. We prune the graph from several hundred to about 100 nodes; only the several most important ones are shown in the figure.

We are currently working on a method to measure the quality of constructed semantic graphs. Both constructing a "golden standard" graph and comparing a given graph to it seems infeasible, so we will most likely resort to evaluating separate stages: triplet ranking, redundant triplet removal and coreference identification, i.e. collapsing nodes.

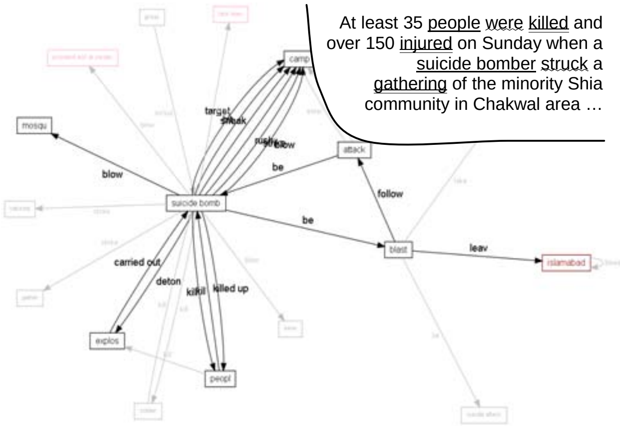

Figure 2. ***A story graph.*** *A sample story graph as constructed by the algorithm. Only the highest scoring nodes are displayed; mid-scoring nodes are partially faded out. In corner, a text snippet from an article on this story is overlaid; subject-predicate-object triplets are marked as output by the tagger. The annotations are linguistically not completely correct but serve our purpose well.*

4.4 Grouping similar stories

We want to identify as an event template every **subgraph** which appears in a convincingly high proportion of story graphs for a set of topically related stories. We consider the template to be a subgraph of a story graph if the story graph either contains its exact copy **or** if the story graph contains a *specialization* of the template graph. A specialization of a graph is an isomorphic graph where one or more node or edge labels have been replaced with more specific terms or synonyms, e.g. "Barack Obama —talk— Ryan Stiles" is a specialization of "politician —discuss— person".

Before we attempt to generate such subgraphs, we must **cluster story graphs** into groups of topically related stories. At the moment, this is done using simple bag-of-words features. To increase the utility of the resulting clusters for template detection, we found it useful to weight all verbs with a factor of 2 (as nouns are more likely to be replaced by their generalizations in the template graph) and to altogether disregard all named entities for the purposes of clustering. For the construction of bag of words vectors, we also use stemming and a stopwords list and prune the vectors to at most 1000 dimensions. Bisecting k-means with cosine distance is run and the resulting clustering hierarchy is cut at a predefined dissimilarity value.

As already mentioned, the algorithm is currently being tested on a sample of about 7000 articles. A completely random sample of articles would cover too many topics, none of which would be sufficiently richly represented for the algorithm to deduce an event template. Therefore, we have augmented the article set with about 1500 articles all reporting on one of three topics we felt were well represented in news: bombings, court sentencings and politicians' visits.

As the purpose of this phase of clustering is to group stories of the same event type (which we interpret as sharing subgraphs of their semantic graphs), it would make more sense to cluster semantic graphs, not bags of words. Unfortunately, this is computationally prohibitive as the clustering has to be fuzzy: the subgraphs burglar-stab-officer and man-shoot-Lennon, for example, both fit the same template but are syntactically completely different. As a compromise between clustering with bag-of-words and graph features, we tried clustering with bag-of-triplets (each triplet is a feature). Contrary to our expectations, this performed much worse than bag-of-words, probably due to data sparsity. We tried alleviating this with latent semantic indexing, but it did not help sufficiently.

4.5 Event template extraction

We observe each cluster of stories separately and hope to extract an event template from it. First, **each node from each graph is** expanded into a *hypernode* – a collection of nodes, at most one per story graph, that best match the given seed node. The matching is computed based on string similarity, WordNet, and GATE entity type (person / organization / location; for example, "Baghdad" and "Kabul" should both fit into a single hypernode as they conceptually play the same role in the bombing template we want to discover). Hypernodes are scored according to their support (how many story graphs contribute a node to the hypernode), coherence (how well the contributing nodes match each other) and importance (average weight/score of supporting nodes in their respective story graphs).

Out of each of the several highest-scoring hypernodes we now try to **grow the template graph**. Starting with a single hypernode, we consider all neighbor hypernodes and rescore them on the basis of their original score and the coherence of the hyperedge with which they would connect to our template-graph-in-the-making. We greedily select the highest scoring neighbor, attach it to the template graph and iterate until the highest scoring neighbor is scored lower than some threshold value.

One last thing that remains to be done is to **generalize (lift) the hypernodes**: at this point, they are only a collection of nodes from concrete stories. Hypernodes with many named entities are generalized into the prevailing entity class name (e.g. "[LOCATION]" in Figure 3). Other hypernodes are generalized into the most specific WordNet synset which generalizes at least half of the contributing nodes. Such a generalized graph is our final result.

The growing and generalization process described in the last two paragraphs is repeated with several different initial hypernodes; whenever the resulting template graph has more than one node and is different from the graphs already generated, we output it.

5 PRELIMINARY RESULTS

The pipeline in its present form is not effective enough to process the very high number of articles needed to obtain a decent number of event templates. There are plans to change that in the near future – for example, triplet extractors much faster than the one we use exist, and exploiting that should speed up the whole pipeline considerably. Even so, the evaluation of such a long pipeline working with large amounts of data is tricky; a proper amount of thought and time should be invested into it. Until then, sample outputs of the algorithm will have to speak for themselves. One of them, the bombing, has already been presented in Section 4. Figure 3 shows the final output of another story cluster, this time on the topic of court sentencings. The template graphs in the figure were extracted from about 10 story graphs each.

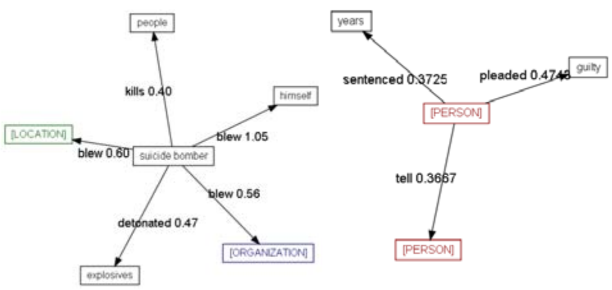

Figure 3. ***The end result.*** *Two event templates as output by the algorithm. The left graph attempts to provide a template for stories on bombings, the left one for stories on court sentencings.*

6 DISCUSSION AND FUTURE WORK

The results, although sketchy, show promise for using the template graphs in ontology extension. Had we used some ontology other than WordNet in the last step, we would essentially get information encoded in the terms of that ontology. While mapping English words to ontology concepts is in general hard, this problem is mitigated by the high redundancy of information found in a collection of news like ours. Each hypernode of our template graph is represented by a whole set of words and therefore easier to interpret in an automated fashion.

In a similar vein, information extraction based on such templates should be feasible as well, since each hypernode is again equipped with a context and a list of words which we can think of as positive examples.

In the future, we hope to be able to verify these claims; in the short run, however, the focus will be on increasing the performance of each pipeline phase.

In the data annotation phase, the use of a faster triplet tagger is a mandatory improvement as the rate of tagging is currently about 2 articles per minute. For named entities we plan to replace ANNIE with a disambiguator proposed in [4] which uses public knowledge sources including DBpedia and GeoNames to tag entities with higher accuracy and using globally consistent IDs.

The clustering of articles into stories will probably be left in Google's domain as its performance is not problematic, although we do have an equivalent in-house solution in store. When scoring triplets at the story level, we might try to exploit the local topology of each article's semantic graph as demonstrated in [5], although statistics alone currently seem to suffice. All in all, the added structure carried by the graphs (as opposed to plain words) will have to be better exploited on all fronts. At this point our assumption that nodes and links of semantic graphs correspond directly to subject-verb-object triplets in English language may prove to be too strong. Indeed, this is not at all always true: for example, for the sentence "neighbors have reported to have seen the car crash into building", parsers would return "neighbors reported car" or similar. The real information, "car crash building", remains hidden deep within the parse tree. With intransitive verbs, even improving the parser would not help: e.g., for "Michael Jackson died quickly", sensible graph representations like "MJ —become— dead", "death —happen— quickly" have no foundation in triplets as there are no triplets at all in the sentence. Both problems are mitigated extensively by redundancy: it is highly probable that some article will use a phrase that the pipeline can recognize, like "Michael Jackson suffered a stroke". If this proves not to be enough, there is interesting work by [9] which aims to syntactically break problematic sentences like the ones above into more parser-friendly but equivalent sentences.

We are also considering altogether dropping the phase of story clustering and trying to mine frequent subgraphs in all the stories. Computational complexity is an obvious issue here, especially because the subgraph support can be fuzzy.

Finally, the most obvious shortcoming of our work so far is the absence of efficiency measures. As the speed and accuracy of the pipeline increase, it will also become feasible to execute larger and more structured tests to properly evaluate its performance.

7 ACKNOWLEDGMENTS

This work was supported by the Slovenian Research Agency and the IST Programme of the EC under PASCAL2 (IST-NoE-216886), ACTIVE (IST-2008-215040) and VIDI (EP-08-01-014) .

References

[1] H. Cunningham, K. Humphreys, R. Gaizauskas, Y. Wilks, "GATE: a General Architecture for Text Engineering," Proceedings of the 16th conference on Computational linguistics, 1996.

[2] D. Klein, C. D. Manning, "Accurate Unlexicalized Parsing," Proceedings of the 41st Meeting of the Association for Computational Linguistics, 2003.

[3] D. Rusu, B. Fortuna, M. Grobelnik, D. Mladenic, "Semantic Graphs Derived from Triplets with Application in Document Summarization," Proceedings of the 11th International Multiconference "Information Society - IS 2008", SiKDD 2008.

[4] T. Štajner, M. Grobelnik, "Story Link Detection with Entity Resolution", presented at WWW 2009 Workshop on Semantic Search, 2009.

[5] J. Leskovec, M. Grobelnik, N. Milic-Frayling, "Learning Sub-structures of Document Semantic Graphs for Document Summarization," Proceedings of the 7th International Multi-Conference Information Society, 2004.

[6] A. Arasu, H. Garcia-Molina, "Extracting structured data from Web pages," Proceedings of the 2003 ACM SIGMOD conference on Management of data, 2003.

[7] H. Tanev, B. Magnini, "Weakly Supervised Approaches for Ontology Population," Proc. of EACL-2006, 2006.

[8] S. Brin, "Extracting Patterns and Relations from the World Wide Web," in Lecture Notes in Computer Science, 1998.

[9] A. Hickl, "Weakly Supervised Approaches for Ontology Population," Proc. of 22nd Conference on Computational Linguistics Coling-2008, pp. 337-344, 2008.

[10] R. Ghani, K. Probst, Y. Liu, M. Krema, A. Fano, Text Mining to Extract Product Attributes, SIGKDD Explorations 2006.