

Learning Event Patterns from Text

Mitja Trampuš, Dunja Mladenič
 Jožef Stefan Institute, Slovenia
 {mitja.trampus,dunja.mladenic}@ijs.si, http://kt.ijs.si

Keywords: information extraction, event mining, pattern detection, semantics representation

Received: [Enter date]

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. Such a template encodes the most characteristic information for a certain type of event; for instance, an earthquake template would encode "x people dead" and/or "town y shook at time z". Event templates have potential to 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.

Povzetek: Predstavljen je cevovod za samodejno ekstrahiranje dogodkovnih predlog (event templates) na podlagi obsežnejšega korpusa časopisnih novic. Dogodkovna predloga je za računalniku prijazna semantična podatkovna struktura, v našem primeru graf, ki nosi vse najbolj tipične informacije za določen tip dogodka. Kot primer: za tip dogodka "potres" si želimo predlogo, ki bi pomenila informacije tipa "x smrtnih žrtev" in/ali "mesto y se je streslo ob času z". Dogodkovne predloge so potencialno uporabne v različnih nalogah ekstrahiranja informacij ter za avtomatske razširitve ontologij. V članku predstavimo uvodne rezultate, dobljene z uporabo cevovoda na testni množici novičarskih člankov.

1 Introduction

Giving rich structure to written natural language is a long-standing goal in the area of text mining. The reason is clear: the amount of information we can extract from the data using shallow approaches like bag-of-words is limited. By enhancing text with structure, we are no longer limited to only statistical approaches, but can start to exploit the broad range of semantic methods.

One of the main approaches toward structuring text 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 Pipeline Overview

To identify event templates mentioned in the previous section, we propose a pipeline working on the *semantic graph* representation of text. A semantic graph is a graph in which nodes represent actors or objects (e.g. "New York" or "computer") and the links between them represent actions (e.g. "accuse" or "visit"). The templates

we are aiming to identify are small subgraphs of these semantic graphs. Because each template should match a whole range of similar texts, e.g. all earthquake reports, templates are not subgraphs of existing semantic graphs in a strict sense; instead, the labels of their nodes are abstracted so that the template contains "city" rather than "New York". See Figure 2 for an example of a semantic graph and Figure 3 for an example of an automatically constructed template.

The templates are constructed by creating a semantic graph for each concrete event reported on by the media and then searching for subgraphs of these graphs such that there are several additional subgraphs in the dataset which generalize to the same template as the observed subgraph. An event template is therefore the most specific graph which still generalizes a sufficiently large number of subgraphs of semantic graphs.

Additionally, each node of a graph representing an event template is rich with statistics about the context within separate articles it appears in, which should future serve as a good starting point for training information extraction methods.

To test the proposed approach, we have used news articles obtained from 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.

3 Processing Steps

As a preliminary, 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* describing a certain *event type*.

For example, the *event template* describing the *event type* of bombings in general may be supported by a *story* of a suicide bomber 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.

In Information Extraction terminology, our event type is a scenario and our event template is a scenario template.

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.

See Figure 1 for a high-level overview of the pipeline. Articles are first obtained from the internet, cleaned and preprocessed; this includes annotating the text with its parse structure, named entities and some more. The articles are then clustered into groups reporting on the same story. The redundancy ensuing in each of these clusters is exploited to identify the most relevant information and represent it as a semantic graph for each cluster. This is followed by another round of clustering, this time on the semantic graphs; graphs that are likely to cover the same event type are clustered together. On each of these clusters of graphs, an iterative process is run to identify event templates, i.e. subgraphs with high support within the cluster. The support is measured fuzzily with the help of WordNet, as two subgraphs talking about related concepts (e.g. "Obama" and "Berlusconi") are said to support each other if they can both be abstracted to a common concept (e.g. "politician").

3.1 Obtaining and Preprocessing Data

Our information source of choice is the newswire. We obtain a selection of articles from the Google's news site <http://news.google.com> by crawling it approximately once every 40 minutes. The site provides as with 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 of the algorithm is to traverse the DOM tree and extract the first block-level element (we have limited ourselves to TD or DIV) containing a lot of text and very little of anything else, particularly links and images. This approach successfully identifies the body of an article with accuracy of above 90%.

Another heuristic has been employed to detect the title of a page. This one is more complex, though the basic idea remains simple: take the first text-only element just above what was previously identified as the content. The result of this heuristic is then combined with contents of would-be <title> and <h1> tags. The accuracy for title extraction is around 90%. While the title does not contribute much information compared to the full content, it does help identify the keywords and eases debugging. Additionally, it is useful for other text mining tasks.

In the end, some additional cleanup is performed; for example, we normalize the encoding, whitespace and punctuation.

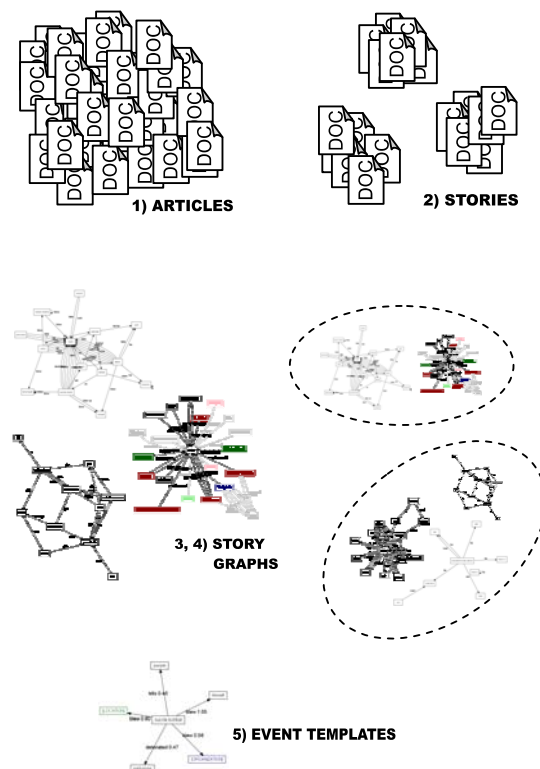


Figure 1: The pipeline step by step. We first preprocess the articles (1) and group them (2) according to the story they cover. A semantic graph is constructed for each story (3). Topically related story graphs are clustered (4); the largest subgraph common to most of the graphs in each cluster (5) is the event template.

3.2 Linguistic and Semantic Tagging

We proceed to enhance the data by tagging it with some semantic information and aligning it to an ontology. Using the ANNIE tool from the GATE [1] framework, we first **detect named entities** and tag them as person, location or organization. Following that, we use the Stanford parser [2] to **extract subject-verb-object**

triplets; the authors report the precision and recall of this stage to be about 85%.

We then 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).

As a last step, we align all triplets to WordNet; that is, for each subject, verb and object appearing in any of the triplets, we try to find the corresponding concept (or "synset", as they are called) in WordNet. The intention is to canonicalize the information representation somewhat, rather than catch all the nuances in the data. Therefore, the alignment strategy can be quite crude: given a word or a phrase, we search for synsets for which this word is one of the known textual representations. If more than one match is found, we choose the one with the lowest ID as WordNet tends to assign lower IDs to more common concepts. The matched synset is declared to be the WordNet representation of our word.

We acknowledge that the triplets acquired in this way do not necessarily correspond to semantic triplets which describe the data. The discrepancies go both ways:

- We include some triplets which do not make sense semantically, e.g. "solution ... be ... proffered". However, this is not overly problematic since they still convey some information.
- We fail to include information which is not encoded with transitive verbs. For example, the can capture the information contained in "The president visited China to ..." as it contains a subject-verb-object triplet, but not that in the semantically equivalent sentence "President's visit to China ...". This shortcoming is alleviated by using redundant information – each story, e.g. president's visit to China, is described by several articles which increases the probability that at least one will convey this information in a form we can detect.

3.3 Constructing Story Graphs

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. As an alternative, we could use simple k-means clustering with cosine distance on the bag of words, which is reported to give reasonable results [13], especially when named entities are weighted higher than normal words. In the case of a large dataset, we could employ a streaming variant of clustering as proposed in the same piece of work. 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 usually 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

other) and importance (average weight/score of supporting nodes in their respective story graphs).

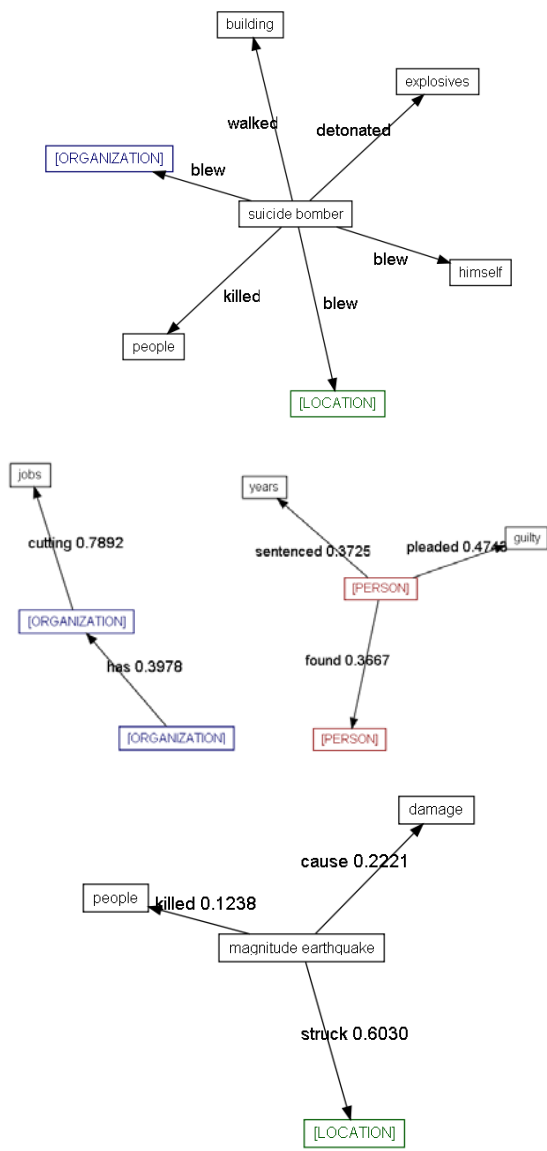


Figure 3: Examples of event templates identified by the algorithm. From top down and from left to the right, the graphs attempt to provide a template for stories on bombings, company layoffs, court sentencings and earthquakes, respectively. Number on the edges are confidence scores.

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.

4 Preliminary Results and Evaluation

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. This problem is accentuated by the high diversity of news topics offered by Google News: to obtain a moderately large number (~20) of stories on e.g. earthquakes, we would have to download literally millions of other stories if we were to sample uniformly.

At the current stage, the pipeline is implemented with no consideration for speed – in a scripting language, with inefficient data structures and algorithms. It is therefore too slow to efficiently process large quantities of data, but does allow for huge improvements. While a faster pipeline is being implemented, we have helped the process by manually interfering with the selection of stories from Google. We have obtained and processed some 7000 articles; 50% of those were chosen by hand to fit one of a few predefined event types (bombing, court sentencing, earthquake, award ceremony, and several more) and the remaining 50% were chosen randomly to introduce more realistic noise.

The quality of the algorithm can be evaluated in two ways: first by separately considering its constituent parts and then by evaluating the outcome of the pipeline as a whole. In the first phases of the pipeline, we use existing or less complex methods and therefore report on their performance only briefly in the relevant sections.

For the HTML markup cleaning step, we define a successfully extracted clear text to be one which does not differ from the golden standard by more than two sentences. With this definition, we achieve precision of above 95% and recall above 90%.

In the named entity detection and annotation step, the existing method achieves recall and precision of about 85%.

For the triplet extraction we set as the golden standard all subject-verb-object triplets. Both precision and recall are low: 40% and 45%, respectively. Note that although this is indeed a problematic part of the pipeline, the figures are particularly low because we require a perfect match with the golden standard which can identify very complex subjects and/or objects (e.g. "the extreme edge of the responsibility to protect"). It has not been determined what percentage of total information is captured with transitive verbs.

When creating the semantic graphs, we evaluate the pruning and scoring step only informally: the method is precision-oriented and it is clear from experimental runs that no important data gets pruned away. Another important aspect of graph creation is coreference resolution, i.e. representing all mentions of a single entity with a single node in the semantic graph. The anaphora resolution method we employed achieves precision of 35% (and perfect recall – it tries to resolve everything).

To roughly evaluate the story graph clustering step, we introduced 6 predefined clusters (a cluster being a set of stories on e.g. earthquakes) with 50 stories among them into a uniformly sampled subset of about 150 stories from Google News. Of the six predefined clusters, three were identified correctly, two were split across two clusters each, and one (on awards, which covers both media and sports awards) was scattered.

We did not evaluate the last step, extraction of patterns, separately, as this step is specific to our application and not relevant as a standalone method.

To evaluate the performance of the pipeline as a whole, we can measure the quality of each separate pattern ("small-scale") as well as the performance of the algorithm on the whole dataset ("large-scale"). Large-scale evaluation is easier. We do not currently have enough data to come to conclusive results so we have also not yet defined a formal metric, but Figure 3 shows 4 out of 7 templates we were hoping to identify based on input data; a reasonable metric should identify those as good. The remaining three desired templates were either not found or extremely low in quality. Slightly tongue-in-cheek, we could claim $3/7=58\%$ recall and 100% precision.

Small-scale evaluation, however, is problematic. How do we define a good template? We should aim for precision first and only then recall if the templates are to be useful. Precision is easier to evaluate. We can define it as the percentage of the template's nodes that relate to information that is relevant to the predominant topic in a cluster (e.g. bombings). Given knowledge of the underlying topic, a human evaluator can identify relevant parts with relatively high confidence. Looking at Figure 3, at most two nodes out of 18 have questionable relevance, so precision is high ($16/18=89\%$). In contrast to precision, small-scale recall is extremely hard to define and evaluate. How do we define all the information types that a template should contain? We have not tackled this problem yet.

Because the amount of data processed so far is quite low, the figures presented above are inconclusive and should be taken with a grain of salt. We feel it is more informative to inspect the templates presented in Figure 3. The graphs in the figure were created by abstracting about 10 story graphs for each.

5 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 Hickl [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.

6 Related Work

Our templates retain information about their groundings (instances) and are as such appropriate for information extraction. Automatic construction of information extraction templates is already relatively well-researched. Many methods aim to extract a single predefined type of information, e.g. the title of a book. To extract different types of information, different classifiers and training data are needed. Examples of such approaches are [6, 8].

The field has lately evolved into the task of *relation extraction* where the goal is to find *pairs* of items related by a predefined relation. As an example, Ghani et al. [10] mine textual product descriptions to simultaneously identify product attributes and their values. Relation extraction is particularly popular in biomedicine where pairs of proteins in a certain relation (e.g. one inhibits the other) are often of interest.

However, the goal of most existing approaches is to obtain **syntactic** templates for detecting words or phrases of a certain type. 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.

Templates somewhat similar to those we aim to construct automatically and with no knowledge of the domain have already been created manually by domain experts. FrameNet [12] is a collection of templates for the events like "disclosing a secret", "speaking", "killing", "arresting" etc. They focus mostly on low-level events, of which typically many can be found in a single document, be it a news article or not. The project does not concern itself with the creation of the templates, other than from the methodological point of view. There is little support for automatic annotation of natural language with the FrameNet frames.

Extraction of attributes characteristic of an event type at a coarser level (or, almost equivalently, of a general domain) is highly relevant for document summarization as it helps identify the most relevant passages of text. Systems entering the Text Analysis Conference (TAC, formerly DUC) summarization challenge often exhibit some knowledge of the domain,

though it is commonly hardcoded in the application and not presented as a event/domain template.

Another related area of research is Topic Detection and Tracking (TDT), where knowledge of topic characteristics can obviously help improve performance. Note that in the case of our algorithm, the benefits are mutual; see Section 3.4 on how we employ TDT.

In addition to enabling information extraction, we aim to obtain templates that are also useful in themselves, as expressions of general truths, for example for ontology extension. Another distinguishing feature of our approach is that we assume no prior knowledge, except linguistic, and rely exclusively on statistics; in particular, we assume no knowledge of the domains discussed in the news articles. Our approach is completely unsupervised and requires no training data.

The most closely related work that we are aware of is by Filatova et al. [11], who also aim to create event templates. They also try to overcome the limitations of syntactic templating and make a step towards more semantic representation of patterns, opting for templates in the form of parse subtrees. In contrast with our work, they need to identify event types and select documents representative of each event type manually.

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.

Acknowledgement

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 VIDII (EP-08-01-014).

References

- [1] H. Cunningham, K. Humphreys, R. Gaizauskas, Y. Wilks (1996). "GATE: a General Architecture for Text Engineering," *Proceedings of the 16th conference on Computational linguistics*.
- [2] D. Klein, C. D. Manning (2003). "Accurate Unlexicalized Parsing," *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.
- [3] D. Rusu, B. Fortuna, M. Grobelnik, D. Mladenic, (2008). "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 (2009). "Story Link Detection with Entity Resolution", presented at WWW 2009 Workshop on Semantic Search.
- [5] J. Leskovec, M. Grobelnik, N. Milic-Frayling (2004). "Learning Sub-structures of Document Semantic Graphs for Document Summarization,"

- Proceedings of the 7th International Multi-Conference Information Society.*
- [6] A. Arasu, H. Garcia-Molina (2003), "Extracting structured data from Web pages," *Proceedings of the 2003 ACM SIGMOD conference on Management of data.*
 - [7] H. Tanev, B. Magnini (2006). "Weakly Supervised Approaches for Ontology Population," *Proceedings of EACL-2006.*
 - [8] S. Brin (1998). "Extracting Patterns and Relations from the World Wide Web," in *Lecture Notes in Computer Science.*
 - [9] A. Hickl (2008) "Weakly Supervised Approaches for Ontology Population," *Proc. of 22nd Conference on Computational Linguistics Coling-2008*, pp. 337-344.
 - [10] R. Ghani, K. Probst, Y. Liu, M. Crema, A. Fano (2006). Text Mining to Extract Product Attributes, *SIGKDD Explorations 2006.*
 - [11] E. Filatova, V. Hatzivassiloglouy, K. McKeown (2006). "Automatic Creation of Domain Templates," *Proceedings of ACL 2006*
 - [12] J. Ruppenhofer, M. Ellsworth, M. Petruck, C.R. Johnson, J Scheffczyk (2006). *FrameNet II: Extended Theory and Practice.* Available at <http://framenet.icsi.berkeley.edu>
 - [13] B. Novak (2008). *Topic Detection and Tracking in a Stream of Documents.* EngD thesis.