Text Mining for Semantic Relations as a Support Base of a Scientific Portal Generator

Vít Nováček*, Pavel Smrž†, Jan Pomikálek*

*Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xnovacek, xpomikal}@fi.muni.cz
† Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
smrz@fit.vutbr.cz

Abstract

Current Semantic Web implementation efforts pose a number of challenges. One of the big ones among them is development and evolution of specific resources — the ontologies — as a base for representation of the meaning of the web. This paper deals with the automatic acquisition of semantic relations from the text of scientific publications (journal articles, conference papers, project descriptions, etc.). We also describe the process of building of corresponding ontological resources and their application for semi–automatic generation of scientific portals. Extracted relations and ontologies are crucial for the structuring of the information at the portal pages, automatic classification of the presented documents as well as for personalisation at the presentation level. Besides a general description of the portal generating system, we give also a detailed overview of extraction of semantic relations in the form of a domain–specific ontology. The overview consists of presentation of an architecture of the ontology extraction system, description of methods used for mining of semantic relations and analysis of selected results and examples.

1. Introduction

The text mining framework described in this paper is used as a part of PortaGe — an ongoing project aiming at semi–automatic generation of scientific web portals. The generator of scientific web portals is meant as an enhanced extension of the existing tools such as Google Scholar (`http://scholar.google.com`) or CiteSeer (`http://citeseer.ist.psu.edu/`). A typical user is a young researcher or a PhD student that looks for relevant information (knowledge) in a subfield (s)he needs to fathom. The interest in the subject is supposed to be long–term, so the user would be notified about new publications, projects, events, calls, etc. in the field. The information on the domain that forms the scope of a generated portal is contained in the respective ontology.

Such knowledge is very useful even as a bare representation of the domain's conceptual structure. However, it also provides mechanisms for a comprehensive specification of the search context when querying the portal. In PortaGe, the user can restrict the search for documents reflecting certain semantic relations, e. g. limit the output to the documents discussing "context-free grammars" as a "tool-for" "analysis of protein sequences". The implemented framework interlinks individual pieces of such knowledge with lexico–syntactic patterns (like hyperonymy or synonymy) able to identify the relations in the retrieved documents. In Table 1 there are given examples of possible relations from a biomedicine domain.

The rest of the paper is organised as follows. In Section 2. we describe the general principles of generation of scientific portals. The architecture of the system for extraction of semantic relations and ontologies is discussed in Section 3. The techniques we use and some results we have obtained are summed up in Section 4. and Section 5. We conclude the article and sketch our future work in Section 6.

2. Basic Principles of the Portal Generation

The current search engines employ user–specified keywords and phrases as the major means of their input. Digital libraries, such as ACM DL (`http://portal.acm.org/dl.cfm`) or Springer DL (`http://arxiv.org/`), add a detailed metainformation level and are able to find publications of a given author, from a given journal, conference proceedings etc. Nevertheless, these services are not able to relate the information to the context of the search. They cannot evaluate what "relevant" means in a particular case.

PortaGe builds a web portal for a domain given by initial data. In addition to the standard keywords, known authors, journals, conferences or projects characterising the subject field, the user can provide seed documents and conference/project web pages relevant for the current search and select apt nodes in the current ontology (automatically extracted from the given and retrieved documents). The tool combines responses from several information sources, such as:

- articles and papers found in digital libraries (ACM DL, Springer Link);
- information from freely accessible web services (arxiv.gov) and web in general;
- local CiteSeer-like publications' database.

The essential component of PortaGe is a WebCrawler focused at common scientific publications' sources, such as conference web pages and homepages of authors. The primary goal of the WebCrawler is to collect publications and to feed the local database by them. Apart from that, it extracts metadata from selected web pages' types, e.g. author's name and contact information from homepages, important dates from conference web pages, etc. The crawler

type of the relation	subject	object	relevance
used_for	SCFG	RNA secondary structure prediction	0.66
described_in	CKY algorithm	Cocke-Kasami-Younger	0.81
is_a	ribosomal frameshifting	RNA function	0.73
abbr_means	HMM	Hidden Markov Models	0.69
abbr_means	SCFG	Stochastic Context-Free Grammars	0.62
is_a	RNA	molecule	0.45
is_a	protein	molecule	0.45

Table 1: A fragment of a sample ontology from bioinformatics texts

is designed to stick to the publications' sources and to ignore any other parts of the web. This makes the crawling very efficient.

PortaGe also employs a collection of components for processing of downloaded publications before they are added to the database. One of these components attempts to extract metadata from each publication, such as the title and list of authors. This process is supported by using information from reference web pages returned by the WebCrawler. Consequently, a list of references is extracted from the new publications. Another component is responsible for matching the references to the publications in the local database. As a result we get a CiteSeer-like database of interlinked publications with basic metadata.

Advanced machine learning techniques are used for classification of publications into relevant portals (based on their texts). The interlinking of publications contributes to the classification significantly. It is very likely that the publications relevant for a given portal are frequently referenced from the publications already stored in the portal's database and vice–versa.

Keywords extraction algorithms are used to obtain the terminology for each portal. This is both valuable information for users and mechanism that finds new possibly relevant publications in the digital libraries by using the keywords search.

PortaGe helps to generate portals not only for individual users; it can support multi–user environment as well. For example, imagine a typical scenario of a team leader that supervises several PhD students. (S)He creates a general web portal that covers various subfields of the area in focus. Individual students work on their particular topics, interact with the system and extend its coverage in the given subfield. Another role of the extracted semantic relations deals with the personalisation of general portals. The system uses the acquired knowledge to evaluate what "relevant" information means for a particular user. Based on user profiles PortaGe defines rules to identify "the best" information for an individual user. A novice (in the given research domain) can ask for introductory documents, others prefer new information (the documents that appeared/were found in the last month), need a general summary of used methods (usually the most referenced documents), or focus on the relevance only.

3. Ontology Acquisition Framework

Our novel knowledge extraction system is called OLE, which stands for "Ontology LEarning". In this section, demands on and architecture of the OLE implementation are discussed.

3.1. Design Considerations

The design of OLE has been influenced by the need for autonomy, efficiency and precision of the resulting platform. The tool should support interactive way of ontology acquisition, but also the fully automatic process of knowledge mining that can run without any human assistance. The efficiency of ontology acquisition is crucial, for the system will process gigabytes of data. The precision is preferred over the recall. Even if the number of the extracted conceptual structures will be relatively low (compared to the number of relations a human can identify in the same resource), it will be balanced by the extensive quantity of resources available.

The relations between concepts stored in the resulting ontology need not be precise —the explicit uncertain knowledge representation is one of the essential parts of OLE reasoning tools to be devised. The increased fuzzy precision of the whole process will balance the loss of exactness.

3.2. System Components

The modular architecture of OLE is given on Figure 1.

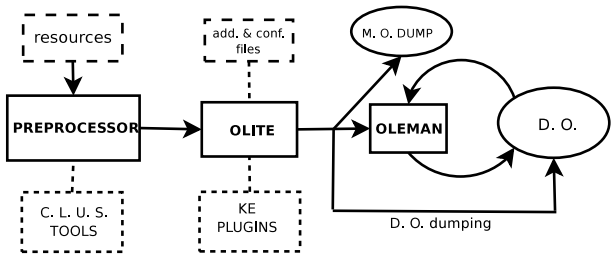

Figure 1: The architecture of the OLE platform

The OLE modules process plain text and create so called miniontologies (that correspond to the particular natural language resources) from the extracted data. Miniontologies can be directly dumped then or passed to the OLEMAN integration module. Here are comments on the figure above:

- **Resources:** relevant documents provided by external tools (document classifiers, existing databases of related resources etc.).
- **C. L. U. S. Tools:** cross–language universality support tools that allow OLE to preprocess data in a given language under specific conditions.
- **OLITE:** the core of the extraction module responsible for creation of ontologies according to provided preprocessed data, utilising some of the knowledge

extraction plugins. The ontologies are primarily produced in a special internal format, but they can be translated into the OWL format (Bechhofer et al., 2004) as well.

- **KE Plugins:** knowledge extraction plugins implementing various methods for miniontology or even whole domain ontology generation.
- **Add. & Conf. Files:** additional and configuration files (such as a file containing explicit patterns for pattern–based extraction method that is described below).
- **M. O. Dump:** direct dump of miniontology as a product of OLITE module.
- **OLEMAN:** module that merges the miniontologies resulting from the OLITE module and updates the base domain ontology.
- **D. O.:** continuously updated domain ontology.

4. Methods Employed for Extraction of Semantic Relations

In the following sections we describe preprocessing and two extraction algorithms that have been incorporated in OLE so far[1].

4.1. Preprocessing of the Natural Language Text

The preprocessing of an input resource has several phases (based usually on fast regular expression matching abstractions):

1. splitting the raw text into sentences, elimination of irrelevant sentences (applied only for certain extraction methods);
2. tokenization and domain dictionary creation (mapping of words to numerical IDs, among other things);
3. POS tagging (based on an *ad–hoc* enhancement of Brill transformation–based algorithm);
4. chunk parsing focused mainly on proper chunking of noun phrases (or possible extraction of pairs of relevant tokens with respective vector–quantised contexts).

4.1.1. Language Universality

The only phase which is strongly language dependent in ontology acquisition is the preprocessing of data. If we can preprocess a text in the steps described above, we can also extract the relations and concepts. On Figure 2 there is a scheme of the tools ensuring language independence of OLE under some conditions. These conditions are mainly determined by existence of respective corpora for the given language.

[1]Note that because the automatic acquisition of conceptually valid ontologies with complex relational structure is often considered as a very difficult or even infeasible task, we have concentrated mainly on taxonomy structure extraction so far. But the techniques used can and will be straightforwardly extended even for extraction of other arbitrary relations.

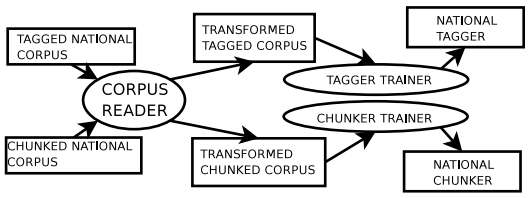

Figure 2: Language universality support

The components depicted at the figure are described as follows:

- **Corpus reader interface** implements sets of rules that transform an existing corpus of particular language into the format comprehensible by OLE tools.
- **Tagger trainer** utilises a tagged corpus in order to create a respective POS tagger; stochastic and Brill transformation–based taggers are trained to be combined with manually designed regular expression and default taggers that are characteristic for the current language.
- **Chunker trainer** utilises a treebank–like corpus in order to learn how to chunk the input tagged sentences and identify particular phrases in a text. However, it is very useful to combine the basic rules learned this way with few additional (manually designed) rules that comprise some OLE–specific phrases (related for example to the pattern–based extraction —see the following section).

4.2. Extracting Taxonomy Parts Using Patterns

The pattern–based method is based on the work presented in (Hearst, 1992). Supposing we know patterns characteristic for a semantic relation (namely the *is-a* relation), we can extract the reference words of concepts involved in the relation from a text.

Within our system, concept extraction utilises the abstract regular expression matching — it works on the chunk–parsed sentences and the compiled PES patterns[2]. The abstract matching means that the objects are not compared as standard strings. They carry information on what are they representing (a chunked sentence or a PES pattern) and what kind of operations should be applied. We have concentrated on creation of taxonomies so far. The taxonomic relations hold especially for nouns. Although we could imagine hyperohyponymic strings for other parts of speech, extraction of the noun hierarchy is the only reasonable way of how to obtain an ontology skeleton. The taxonomy pieces are extracted directly according to the respective patterns.

We focus at noun phrases appearing in the vicinity of so called core phrases[3]. The algorithm implemented in OLE is non–formally described as follows:

[2]Stands for *Pattern Extended Specification*, based on sets of various regular expressions, universally encoded in XML format.

[3]Static words that always appear in a pattern, thus identifying it in a text.

input: chunk–parsed sentence
output: list of *is-a* relations

1. for each matching pattern within a sentence, create a triple(s) (l, c, r), where l, c, r are left context, core phrase and right context respectively;
2. for each triple, get the complex noun phrases that are nearest to the core phrase;
3. assign the hyperonymy/hyponymy mark for these phrases according to the respective matching PES pattern object;
4. extract the hyperonyms/hyponyms from each noun phrase NP in the following manner:
   - 4.1 if NP is a simple noun phrase, extract the head noun from the phrase and return it as single hyperonym/hyponym;
   - 4.2 if NP is a conjugated noun phrase consisting of several simple noun phrases, extract the nouns or noun compounds form each one of them and return them in hyperonym/hyponym list;
5. join all the hyperonyms/hyponyms gained in the step 4. in respective *is-a* relation instances;
6. return the relation instances;

The extracted information is stored in a universal internal format. An output miniontology file can be produced by applying respective translation rules. However, the miniontology is primarily passed to the integration phase. We have employed a very naïve distance–similarity technique for this merging so far, but we plan to use much more efficient novel uncertainty reasoning methods in future[4].

4.3. Extracting Taxonomy Using Hierarchical Clustering

We use the hierarchic clustering and consequent WordNet–based[5] annotation of resulting classes in order to acquire sets of individuals and their classes' hierarchy, which naturally corresponds to an ontology taxonomy. The approach is somehow similar to work presented in (Pantel and Lin, 2002) in the context of automatic extraction of senses from a text, although we rather discover relations between groups of words than their senses. Because of extensive exploitation of the provided data (all present nouns are taken into account), we use the method to create the initial domain ontology.

The hierarchic clustering of words is described for example in (Ushioda, 1996). It is based on iterative clustering of similar words and then their clusters in a tree structure, which is called a "dendrogram". Various similarity measures can be used —we use the cosine distance of normalised vectors that represent contexts of respective terms.

4.3.1. Algorithm Description

By the hierarchical clustering we obtain classes of words, classes of these classes of words and so forth until we generally have the least specific root of the respective tree (called a dendrogram in the field of machine learning). The history of merging steps allow us to reconstruct a noun taxonomy from the dendrogram, but the classes at particular levels are anonymous. If we want to translate them into an ontology, we have to annotate them by names that conform to hyperonymy relation in the context of words comprised by the respective class. The bottom level (dendrogram leafs) represent individuals in the ontology then, as the upper levels represent classes. We do this more or less empirically, using the WordNet lexical database. We call this novel technique CAANNO (Clustering and Autonomous ANNOtation). The algorithm for CAANNO acquisition of a named classes' hierarchy is described as follows:

input: vectors corresponding to nouns associated with their contexts
output: internal representation of an ontology taxonomy

1. induce a dendrogram from the preprocessed input data;
2. for each level in the dendrogram tree (going from the bottom to the root), perform the following:
   - 2.1 to all words in each class at the current level, assign a set of all hyperohyponymic strings in WordNet they are involved in;
   - 2.2 compute the most frequent hyperohyponymic string among each class;
   - 2.3 from the nearest hyperonym in this string (that is not present as an individual in a class), make an annotation for this class (take an arbitrary word from the respective synset and update the domain dictionary if needed);
3. translate the annotated tree into an ontology representation;

The step 2.2 in the above algorithm forms a computational bottleneck for autonomous class annotation —we should mutually compare all hyperohyponymic string with respect to all senses the involved words can have. This causes very undesirable combinatorial explosion.

Therefore we use a heuristics to overcome this problem. It is very simple —when computing the string frequencies among a class, we process only strings corresponding to all senses of all words in a class in one pass. The addition of each string that is corresponding to a word w is weighted by the $\frac{S_{avg}}{S_w}$ modifier, where S_{avg} is average number of strings per one word in the given class and S_w is number of strings for the word w.

Such weighting corresponds to an intuitive idea that the words with more hyperohyponymic strings assigned should not overwhelm the words with less strings in the frequency computation. This heuristics solves the combinatorial explosion problem and yet it does not reasonably harm the annotation usefulness, as seen in Section 5.

5. Preliminary Results and Examples

In the following sections we present selected preliminary results of the ontology extraction and merging techniques that have been used in OLE. The evaluation of an ontology is a problematic task even for hand–crafted ontologies. As stated for example in (Brewster et al., 2004), the well known notions of precision and recall can hardly be used. Following the source cited, we would like the *precision* to reflect the amount of knowledge correctly represented in an ontology with respect to all knowledge in it. On the other hand, the *recall* should reflect the amount of knowledge stored in an ontology with respect to all knowledge available. It is obviously very hard to define a *correctly represented knowledge*, as well as to decide what is *all available knowledge*. Therefore we cannot perform an exact and exhaustive evaluation similar for example to evaluation techniques used for POS tagging or disambiguation NLP tasks.

[4] These topics are covered by our other works like (Nováček, 2006b) or (Nováček, 2006a).

[5] WordNet is a large lexical–database with rich interrelated structure. See (Fellbaum, 1998) for details.

Due to the complications mentioned in the previous paragraph, we manually analysed representative or illustrational samples of ontologies having been gained from various domain–specific resources. The presented values of precision are orientational ratios of number of relations that were found to be "reasonable" compared to number of all relations in an ontology. The recall values are even more difficult to be specified. Therefore we use only a simple measure —ratio of number of concepts (individuals) contained in the created ontology compared to the number of nouns present in the processed text.

5.1. Selected Results

For the pattern based method, we tested the system with patterns given in Table 2 below[6]. The patterns are presented in common regular expression–like syntax.

Id	The pattern
1	NP such as (NPList | NP)
2	such NP as (NPList | NP)
3†	(NPList | NP) (and | or)other NP
4	NP (including | especially) (NPList | NP)
5‡	(NPList | NP) (is | was)an? NP
6†	(NPList | NP) is the NP
7†	(NPList | NP) and similar NP
8†	NP like (NPList | NP)

Table 2: Patterns for *is-a* relation

Other patterns can be added easily, but the patterns presented in the table were found to be sufficient for basic evaluation.

The average time of processing of a resource (only extraction without miniontology dumping or merging) was 0.94 seconds[7]. Average size of a document in the sample that was used for this performance test was about 66.7 kilobytes (approximately $9,500$ words). This results in a processing speed of about $10,100$ words per second which we find satisfactory.

For the manual evaluation we randomly chose ten resources from the whole document set (about $12,000$ articles from computer science domain in this case). For each miniontology created by OLE system we computed the ratio of "reasonable" relations compared to all extracted relations. For all the measures of informal precision (*Pr.*) and recall (*Rec.*), an average value was computed. We present these results in Table 3, provided with respective average original resource size and number of all concepts extracted (in the *M1* row).

In the same table, there are also similar results of CAANNO clustering–based technique (in the *M2* rows). Due to the strenuousness of manual evaluation of large ontologies we used only a set of 131 concepts (non–unique individuals) from a coherent computer science domain resource. 60 unique individuals and 47 classes were induced. We distinguished between *class–class* and *class–individual* relationships when analysing the precision. We were interested in whole dendrogram structure (with one root and complete history of clustering of all leafs), but it is more convenient to reduce the height of the tree for practical reasons[8].

The CAANNO precision is lower than for the pattern–based method, but it slightly increases with more data. Moreover, the absolute recall is no doubt much higher. Relatively low precision could also be improved by utilisation of reasoning supported by integration of more precise ontologies and even by heuristic comparison of different cuts of a dendrogram for the same resource set.

Method	Res. sz. (wrd.)	No. of conc.	No. of rel.	Pr. (%)	Rec. (%)	I (%)
M1 avg.	4093	22.6	14.5	61.16	1.57	3399.17
M2 cl.–cl.	-	47	99	38.38	100	2183.62
M2 cl.–indiv.	-	60	62	51.61	100	5691.05
M2 sum–up	486	107	161	44.99 (avg.)	100 (avg.)	3937.34 (avg.)

Table 3: Selected results of OLE's taxonomy extraction tools

Nevertheless, precision values for both methods are quite high when we look at the *I* column in the table. The *I* values present an improvement in precision over a base–line, which is computed as $\frac{R_R}{N(N-1)}$, where R_R stands for number of reasonable relations and N is the number of concepts in an ontology[9]. Moreover, it is precision of the extraction phase and it would be significantly improved by utilisation of a proper reasoning engine. This is one of the main goals of our future work.

5.2. Examples of Ontology Portions

The figures below represent illustrational samples from ontologies created by OLE. Classes are depicted as ellipses, individuals as squares and arrows from hyperonyms to hyponyms encode the *is-a* relation.

On Figure 3, there is depicted a sample from one of the thousands of ontologies gained by pattern–based extraction from computer science articles.

The sample of experimental single–rooted ontology extracted by CAANNO technique is on Figure 4.

We tested the CAANNO technique with significantly reduced height of the dendrogram also for resources from the domain of emergency management. Sample from the ontology is given on Figure 5. Further comments on this very interesting experiment are unfortunately beyond the scope of this paper. However, the example shows how even the incorrectly[10] assigned relations can be used as a base for manual or semi–automatic definition of other very useful relations. These are for example the relations between concepts *hospital* and *biologist* or *river* and *peaks* on Figure 5.

[6] † — introduced by author, ‡ — modified by author, others adopted according to (Hearst, 1992) and (Etzioni et al., 2004); however, the devision of simple patterns is quite easy, therefore similar patterns can be found even in other works.

[7] On a machine with 3.2 GHz Intel Pentium 4 processor and 2GB of RAM, powered by Ubuntu Linux operating system.

[8] For the taxonomies in human conceptual structure representations, which are apparently performing quite well, obviously do not exceed a depth of at most tens.

[9] The $N(N-1)$ is number of all *is-a* relations that can be assigned among all concepts.

[10] In the meaning of strict hyperonymy.

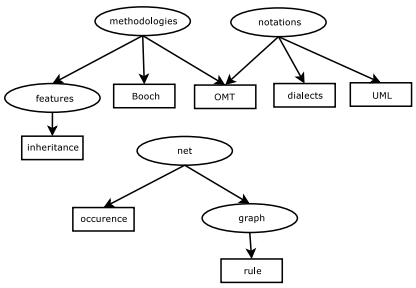

Figure 3: Sample of ontology created by pattern–based method (computer science domain)

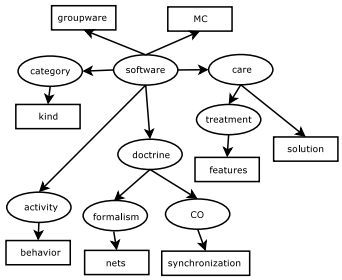

Figure 4: Sample of ontology created by CAANNO method (computer science domain)

6. Conclusions and Future Work

We have presented a description of novel tool for autonomous mining of semantic relations (namely the taxonomic so far) in the form of a domain specific ontology. The system is able to continuously process vast natural language resources, create a knowledge base from them and thus contribute to efficiency of devised PortaGe system for automatic generation of scientific portals.

A lot of work still needs to be done on both tools, the PortaGe system and the OLE framework. Currently, techniques for extraction of another relations are under development. Our further research focuses at the progressive improvements of basic implementation of CAANNO technique and on the design and implementation of advanced mechanism covering uncertainty in the acquired ontologies. We will also work on a qualitative evaluation of the scientific portals generated by PortaGe. They would be employed for example for e–learning of PhD students at our universities, but they could be applied in any domain of science as a very useful speed–up tool for general research in a broader scope.

7. Acknowledgements

This work has been supported by Academy of Sciences of the Czech Republic, 'Information Society' program, the national research project 1ET100300419.

8. References

S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, 2004. *OWL Web Ontology Language Reference*. Available at (February 2006): `http://www.w3.org/TR/owl-ref/`.

C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks. 2004. Data driven ontology evaluation. In *Proceedings of LREC 2004*.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 100–110, New York, NY, USA. ACM Press.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA. Association for Computational Linguistics.

V. Nováček. 2006a. Motivations of extensive incorporation of uncertainty in ole ontologies. In *SOFSEM 2006: Student Research Forum Proceedings*. Institute of Computer Science, Academy of Sciences of the Czech Republic.

V. Nováček. 2006b. Ontology learning. Master's thesis, Faculty of Informatics, Masaryk University, Czech Republic. Available at (February 2006): `http://nlp.fi.muni.cz/~xnovacek/files/dp.pdf`.

P. Pantel and D. Lin. 2002. Discovering word senses from text. In *Proceedings of ACM SIGKDD02, 2002*.

A. Ushioda. 1996. Hierarchical clustering of words. In *Proceedings of the 16th conference on Computational linguistics*, pages 1159–1162, Morristown, NJ, USA. Association for Computational Linguistics.

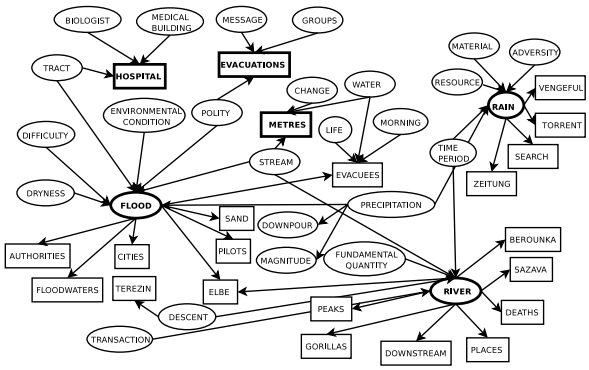

Figure 5: Sample of emergency–management (flood) domain ontology