

A Pipeline for Supervised Formal Definition Generation

Alina Petrova

Technische Universität Dresden
alina.v.petrova@gmail.com

Abstract. Ontologies play a major role in life sciences, enabling a number of applications. Obtaining formalized knowledge from unstructured data is especially relevant for biomedical domain, since the amount of textual biomedical data has been growing exponentially. The aim of this paper is to develop a method of creating formal definitions for biomedical concepts using textual information from scientific literature (PubMed abstracts), encyclopedias (Wikipedia), controlled vocabularies (MeSH) and the Web. The knowledge representation formalism of choice is Description Logic as it allows for integrating the newly acquired axioms in existing biomedical ontologies (e.g. SNOMED) as well as for automated reasoning on top of them. The work is specifically focused on extracting non-taxonomic relations and their instances from natural language texts. It encompasses the analysis, description, implementation and evaluation of the supervised relation extraction pipeline.

Keywords: knowledge representation, non-taxonomic relationships, ontology learning

1 Introduction

Formalization of biomedical knowledge has long been an area of active research. Existing biomedical knowledge resources vary considerably in terms of their formalization principles, from databases and data collections (e.g. MEDLINE¹), to taxonomies and controlled vocabularies (e.g. MeSH²), to proper ontologies with rich formal semantics (e.g. SNOMED³). They also vary greatly with respect to the domains and areas they cover, size, age, ways of maintaining and integrating new knowledge etc. Formally representing the biomedical knowledge can bridge the gap between existing resources and enrich them as well as process the newly generated knowledge that comes in abundance and is publicly accessible.

Research in life sciences is characterized by the exponential growth of the published scientific materials: articles, patents, technical reports etc. MEDLINE, one of the biggest bibliographic databases for biomedicine, currently contains

¹ <http://www.ncbi.nlm.nih.gov/pubmed>

² <http://www.nlm.nih.gov/mesh/>

³ http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

more than 23 million articles. The average amount of newly added articles comprises 15000 items per week. To handle such an amount of information, multiple initiatives have been launched for the purpose of organizing biomedical knowledge formally, e.g. using ontologies [3]. An ontology is a complex formal structure that can be decomposed into a set of logical axioms that state different relations between formal concepts. Together the axioms model the state of affairs in a domain. With the advances in Description Logics (DL) the process of designing, implementing and maintaining large-scale ontologies has been considerably facilitated [1]. In fact, DL has become the most widely used formalism underlying ontologies. Several well-known biomedical ontologies, such as GALEN or SNOMED CT [4] are based on DL. SNOMED CT has adopted the lightweight description logic *EL++* that allows for tractable reasoning.

There are several benefits of formal knowledge representation. First of all, it enables efficient information integration; already existing knowledge about the entity can be aggregated from multiple resources, and the new knowledge can be easily integrated. Secondly, formal knowledge representation makes it possible to automatize a number of crucial tasks that deal with information processing: efficient search, validation and reasoning. Finally, formal representation can support knowledge visualization which itself can bring about further insights about the domain, i.e., facilitate knowledge discovery.

1.1 Two Examples of Biomedical Knowledge Formalization

In this section we present two recent works in which the application of formal ontologies to biomedical knowledge produced interesting results that demonstrate the usefulness and the potential of knowledge formalization in the biomedical domain. In [4] the authors used the Foundational Model of Anatomy (FMA), an ontology of human anatomy, where concepts are linked with the *part-of* relation. They annotated images of penetrating injuries with anatomic concepts, thus disambiguating the visible regions of the body, and then performed logical reasoning over the ontology to predict the possible internal damages caused by the injury. The project was conducted by the U.S. Defense Advanced Research Projects agency and has life-saving importance. [5] ran an even more large-scale and ambitious project. Its aim was to create a robot scientist - a robot that conducts independent research, that is, sets a hypothesis, tests it experimentally and reasons about the acquired data by interpreting the results, all on its own. The developed robot had a rich knowledge base on the backbone that was used at all stages of the research process. The robot was provided with a general biomedical database as well as with a formal model of yeast metabolism, and it autonomously generated and validated experimentally functional genomics hypotheses, thus becoming the first machine that made a scientific discovery without human intervention. The two works illustrate the huge range of applications that formal knowledge resources can have in life sciences as well as their unbounded potential. Not only do they help sustain the ever-growing collection of already published results, but they can also lead to knowledge discovery through formal reasoning.

1.2 What is Formal Definition Generation?

Formal definition generation (FDG) is a type of knowledge modeling that translates a natural language definition into a formal representation using some formal language notation. FDG can be viewed as the automatic acquisition of complex axioms for an ontology. Unlike the taxonomy acquisition, which seeks to identify parent-child relations in text and is usually based on simple patterns [6], FDG focuses on highly expressive axioms containing various logical connectives and non-taxonomic relation instances.

Formal definition generation can be illustrated by an example: a natural language sentence with a classic definitional structure *A is a type of B that has a specific property C* is translated into a formal representation $A \equiv B \sqcap \exists \text{hasProperty}.C$. The formalism of choice here is Description Logic (DL).

Some definitions can be straightforwardly rewritten into a formal language:

Acenocoumarol: a coumarin that is used as an anticoagulant.

It is a definition taken from the MeSH controlled vocabulary. If we assume that *Acenocoumarol*, *Coumarin* and *Anticoagulant* are valid biomedical concepts, the definition can be encoded by means of a simple DL in the following way:

$$\textit{Acenocoumarol} \equiv \textit{Coumarin} \sqcap \exists \textit{used_As}.\textit{Anticoagulant}$$

The encoding is very simple since there exists an almost perfect one-to-one correspondence between the lexical items in the definition and the elements of the formal syntax. However, this is not the case for the majority of the sentences. FDG does not boil down to a mere re-writing of textual definitions using a different notation; instead, it is a complex task that requires thorough analysis and understanding of utterances and their constituents. Below are the examples of MeSH definitions that are far more difficult to process:

Acetolactate Synthase: a flavoprotein enzyme that catalyzes the formation of acetolactate from 2 moles of pyruvate in the biosynthesis of valine and the formation of acetohydroxybutyrate from pyruvate and alpha-ketobutyrate in the biosynthesis of isoleucine.

Lissamine Green Dyes: green dyes containing ammonium and aryl sulfonate moieties that facilitate the visualization of tissues, if given intravenously.

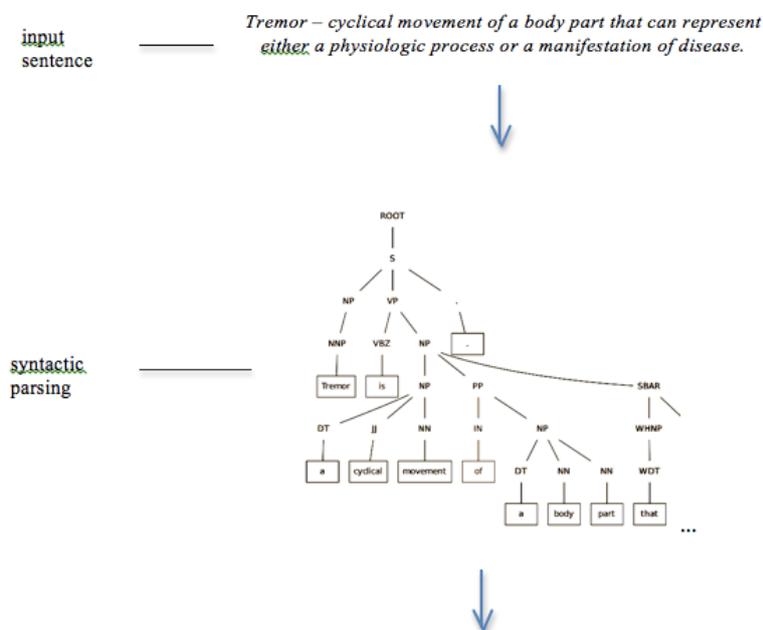
Even definitions for which finding a formal representation appears to be trivial may in fact contain various pitfalls. How exactly should the definition *Acepromazin is a phenothiazine that is used in the treatment of psychoses* be formalized? Should *the treatment* correspond to an independent concept that is linked to *emphAcepromazin* by the *used_in* relation? Or should it rather correspond to the relation *treats* that takes as arguments *psychosis* and *phenothiazine*, and ultimately *acepromazin*? The answer to this question is not obvious and is heavily dependent on the way one chooses to model the knowledge.

2 Methodology

This section describes the pipeline for formal definition generation (FDG) from text. The core step in FDG is non-taxonomic relation extraction: not only expressive relation instances account for the most part of definition formulas, but they also require such tasks as concept annotation and taxonomy detection as preprocessing steps. Hence, the FDG pipeline in essence tackles the task of relation extraction. It consists of the following main steps, illustrated by Figure 1 and discussed in the subsequent sections:

- syntactic parsing of the input sentence;
- semantic annotation of the sentence with biomedical concepts;
- extraction of semantic triples from syntactic paths between the annotated concepts;
- classification of the triples as pertaining to specific biomedical relations;
- final formula generation based on the labeled triples.

Each step is discussed in detail in the subsequent sections. Figure 1 illustrates the pipeline using the MeSH definition of *Tremor*:



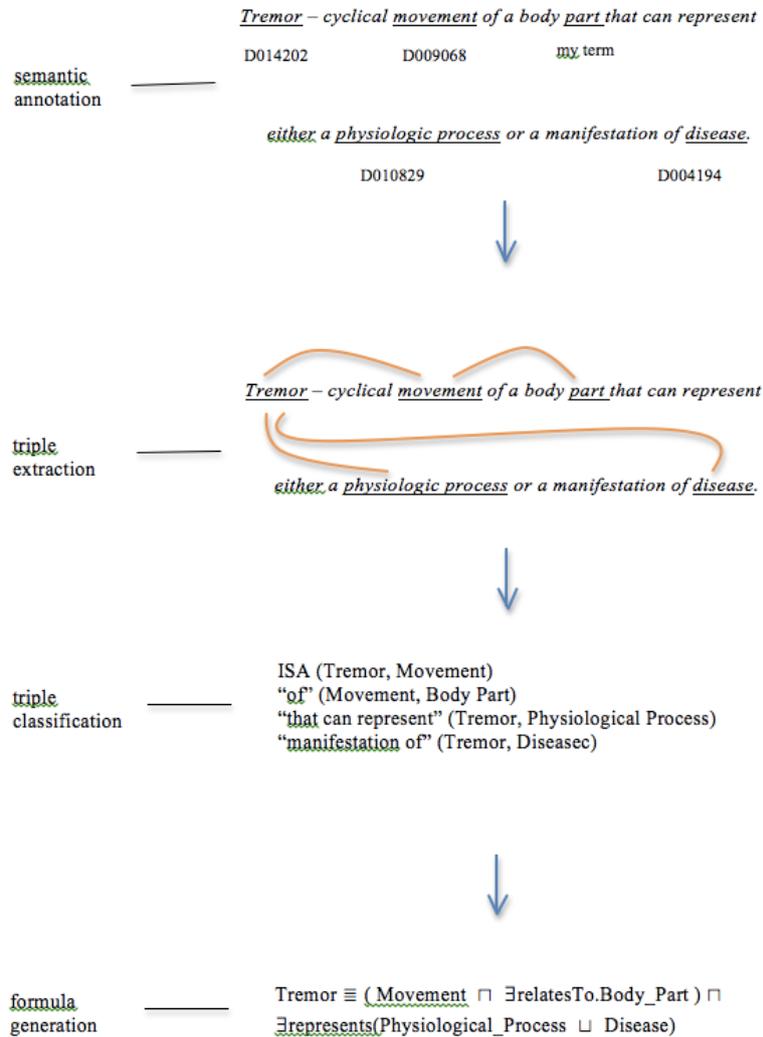


Fig. 1. Formal definition generation pipeline. The definition for *Tremor* is used as an example.

2.1 Parsing and Annotation of Input Sentences

Parsing and annotation can be viewed as two pre-processing steps of the pipeline. They both rely on separate components, syntactic parsers and semantic annotators respectively, enriching the initial textual input with additional information, i.e. the syntactic dependencies and the occurrences of concept mentions in the sentence. In the present work we use external parser and annotator, since their

creation is itself a stand-alone research problem which lie outside the scope of formal definition generation problem.

Given an input text and an ontology that describes the domain, concept annotation, also called semantic indexing or concept recognition, is the task of finding in text mentions of ontology concepts and mapping the corresponding lexical tokens to concepts. Typically, biomedical concept annotators aim at recognizing textual occurrences of diseases, drugs, genes, body parts, species and in principle, any other conceptual entity that exists in the input ontology.

There are multiple third-party biomedical annotators available online. As a rule, they use specific knowledge resources, i.e. ontologies and thesauri, as repositories of concepts they aim to find in texts. One of the most widely used one is MetaMap [7]. It is a dictionary-based system that indexes biomedical text with UMLS concepts [8]. We use MetaMap as the annotator of choice.

2.2 Triple Extraction

After a text string is annotated with biomedical concepts, the next step is to group these concepts into relational instances and to form a preliminary structure of the formal definition. It is the task of the triple extraction component. The parser takes as input a textual definition pre-annotated with biomedical concepts as well as its syntactic parse tree and produces structures of the form *concept_A* – *relational_string* – *concept_B*, which we call *unlabeled triples*.

The triple extraction component runs as follows:

1) **detect the parent term**, if it is present

At this step we rely on the information provided by the ontology used for the semantic annotation: if the term that appears first in the definition is not recognized by the annotator, i.e. it is not considered as concept by the ontology, then it belongs to a relational string of some triple (*Abdominal Wall* example); otherwise it is a parent concept (*Cattle Diseases* example).

Abdominal Wall: the outer margins of the abdomen, extending from the osteocartilaginous thoracic cage to the pelvis.

Cattle Diseases: diseases of domestic cattle of the genus bos.

2) **group coordinated concepts into conjunctive or disjunctive sets**

Detecting coordination is one of the very important issues in predication extraction. Coordinated concepts are organized into sets with one representative concept. Whenever this concept is part of a triple, the rest of the concepts automatically form triples as well, using the same relational string and the same concept as the second argument, e.g.:

Vesicular stomatitis Indiana virus: the type species of vesiculovirus causing a disease symptomatically similar to foot-and-mouth disease in cattle, horses, and pigs.

Foot-and-Mouth Disease — in — Cattle
Foot-and-Mouth Disease — in — Horses
Foot-and-Mouth Disease — in — Swine

Triples for coordinated concepts will further be transformed into conjunction and disjunction of concepts in DL notation of the definition formula.

3) organize the concepts into concept pairs

This is the key step in the definition parsing as it shapes the resulting triples. The process is heavily dependent on the syntactic structure of the sentence. One straightforward way of linking concepts together would be to follow the dependency paths across the syntactic tree and to link every concept with the nearest dominating one (and to link the top concept with the head concept). However, while parsing the definition, we would like to collect as much information about the head term as possible. For this reason we link annotated concepts with the head term whenever it is possible and does not violate the common sense. In fact, for the majority of the triples both ways of constructing triples are possible and comprehensive. For example, in the following definition:

Classical Lissencephalies: disorders comprising a spectrum of brain malformations representing the paradigm of a diffuse neuronal migration disorder,

if *Classical Lissencephalies* is a malformation that represents *Diffuse neuronal migration disorder*, we can induce that it represents a disorder. Thus, by linking concepts occurring in the definition with the main term we skip this induction process.

4) extract relational strings

To finish the formation of unlabeled triples, we need to accompany concept pairs with relational strings that contain the mention of the respective relation in text. The intuitive approach is to take the strings that are located in between the two concepts in the pair. It has two major disadvantages, though: the in-between string might either contain more than just the relation mention and thus cause noise during classification, or it might as well not contain the mention altogether. To avoid such mistakes, we extract only the substring between the current concept and the preceding one, independently of the position of the second concept:

Hypothalamic Hormones: peptide hormones produced by neurons of various regions in the hypothalamus.
Peptide hormones — produced by — Neurons
Peptide hormones — of various regions in — Hypothalamus.

5) detect negation

The negation is detected in the relation string using simple patterns. If the string

contains tokens like not or other than, the triple is considered to be negated. This information is useful and should be propagated till the step of formula generation. After the parsing is completed, the parser outputs unlabeled triples of the form *concept_A* — *relational_string* — *concept_B*, possibly accompanied with the NEGATION mark. The number of triples for a definition depends directly on the number of annotated concepts.

2.3 Triple Classification

The last step of the formal definition generation pipeline takes as input the unlabeled triples generated by the parser and substitutes the relational strings with the relation labels reducing the relation instances to the invariants of some domain-specific relation.

Labeling the text strings with relation names is an instance of the text classification task. Relational instances, or triples, represent the training/testing examples and form the learning corpus. Every instance is represented as a set of features and passed to a machine learning algorithm. The learning instances are labeled with a class — a relation name in our case. The model is then trained using the labeled instances and is used for the classification of new instances that do not yet have a label. We assume that a relational instance corresponds to a precisely one relation, thus the task of relation labeling is a single-label classification.

Thus, the most important step of triple classification is to train such a model that will perform accurately on the new input definitions. The model cannot be trained on the textual instances per se, but rather on their formal representation as sets of features. In this work we extracted two types of features from the relational triples: lexical and semantic features.

Lexical features correspond to relational string of the triple. We used the so-called character ngrams as lexical features. Given a textual instance *I* and a value for the parameter *n*, we now examine *I* as an ordered series of characters instead of words. For the extraction of the character ngrams, we are using a sliding window of size *n*, and we do not exclude space characters in order to capture patterns across token boundaries. For example, for a string *is pneumoconiosis caused by* the character bi-grams are: *pn, ne, eu, um, mo, oc, co, on, ni, io, os, si, c, ca, au, us, se, ed, d , b, by*. Each instance *I* (in our case the text between the two concepts) can be represented as a feature vector, features being ngrams and the value of each feature being 0 or 1, depending on whether a term occurs in the instance (1) or not (0).

Ngrams are able to implicitly capture a huge variety of information about a string. In particular, character ngrams can reflect word order, lemmas, stems and grammatical forms of words, important morphemes, to name a few. All this information is utilized at a cheap cost: no sophisticated linguistic analysis is required for the ngram extraction. Obviously, a single ngram does not play a big role in the labeling process, but several ngrams of the same string taken together can yield a strong signal of a particular class. For example, a set of trigrams *{cau, aus, use, sed, ed, ed ,d b, by}* extracted from a relational string

caused by captures not only the stem form of the verb, but also the fact that it is used in passive voice and is followed by a preposition *by*, which reflects a very common lexical pattern for the causative relation.

Unlike lexical features, **semantic features** account for the argument concepts of the triple. As semantic features, we used concept types of the arguments, i.e. broad semantic categories of concepts. In order to determine the types of triple concepts, one needs an underlying terminology where all the concepts are combined into an ontology. Every concept can be reduced to some upper concept of broad semantics, which will be considered as a semantic type and used as a features. In our work we used the UMLS Semantic Network as the source of types. The UMLS Semantic Network [9] is an upper ontology for the biomedical domain which forms the top level of the UMLS concept hierarchy. It has 133 semantic types and 54 semantic relations. Types and relations are very broad and are used for high-level categorization and interlinking of concepts. Types are assigned to all the concepts of the UMLS, thus the type information is easily accessible.

Thus, every relational instance, i.e. triple, is represented as a set of all character tri-grams extracted from the relational string, and a pair of concept types for the relational arguments. A model then is trained on all processed instances and is used for the classification of unseen instances.

2.4 Formula Generation

When all the triples are extracted from the input textual definition, the last step is to combine them into a formula. In the current version of the pipeline we follow the formalization of biomedical knowledge used in SNOMED CT ontology and quantify all relational instances existentially. All triples are then combined conjunctively.

3 Evaluation and Discussion

As it has been stated at the beginning of the paper, relation extraction is at the core of formal definition generation. Therefore, in this section we discuss the performance of our pipeline with respect to relation extraction. In particular, we separately evaluate the steps of triple extraction and triple classification.

3.1 Triple Extraction

As a preliminary evaluation of the parser, we have run it over a small corpus of 40 definitions and manually annotated the generated triples as correct or incorrect. MeSH serves the source for textual definitions to be parsed. The triples were evaluated as follows: we mark a triple as correct if the two concepts serving as arguments of the relation are chosen correctly and the relational string is also parsed correctly (it does not miss anything). If any of the two conditions was violated, the triple was considered incorrect. For 40 randomly selected MeSH

definitions annotated with 147 concepts from MeSH and from the extended vocabulary the parser generated 110 triples. 98 triples are manually labeled as correct, only 11 triples (10%) are incorrect. In particular, for 32 definitions out of 40 the triples are generated correctly (80%).

3.2 Triple Classification

For the evaluation of relation classification process given the set of features we designed (see section 2.3), we relied on the external corpus. An external corpus is needed to exclude the errors passed from the previous steps of FDG pipeline which would affect the classification performance. We used *SemRep Gold Standard* corpus, which consists of 500 MEDLINE sentences manually annotated with relational triples. The annotation includes concepts, concept types, relational strings and relation labels. The corpus contains 1357 instances of 26 distinct relations. The the top occurring relations are *process_of*, *location_of*, *part_of*, *affects*, *treats*. The corpus was used for training and testing of the SVM classifier using 10-fold cross-validation. The tests were run for top 5, top 10 and for all 26 relations. The resulting F-measure is 94%, 89.1%, 82.7%, respectively. It should be noted, that the top 5 relations account for 63% of all relational instances, which means that with our learning method we can classify the majority of relational instances with an extremely high F-measure of over 90%.

3.3 Related Approaches

One previous approach of Formal Definition Generation is described in [10]. The authors reformulate the task into an automatic acquisition of ontology axioms from natural language texts. The formalism of choice is *SHOIN*, a very expressive DL that is able to model negation, conjunction, disjunction, and quantitative restrictions. The developed system *LExO* is based on full syntactic parsing of input sentences. The dependency tree is transformed into DL formulas through a chain of hand-written syntactic rules that take into account parts of speech, sentence positions, tree positions and syntactic roles of all words. The rules cover a broad set of syntactic structures, such as relative clauses, prepositional, noun and verbal phrases.

Another related approach [11] belongs to the area of ontology acquisition. Ontologies consist of terminological axioms (TBox) and assertional facts (ABox). In this paper, we focus on acquiring a special but common TBox knowledge — formal definitions — from texts. [11] mainly studies the ABox extraction, whereas the enlisted TBox acquisition approaches are mainly based on syntax transformation.

The mentioned systems have several limitations in formalizing the definitional sentences, which stem from their rule-based nature. Two main problems are semantically ambiguous relation mentions, e.g., *of*, and relation mentions with similar semantics, but dissimilar linguistic form, e.g., *Causative_agent* relation in SNOMED CT can be expressed both by *caused_by* and *due_to*. Natural language is versatile and complicated, and the same meaning can be expressed in multiple

ways. Hence, it is not possible to cover all ways of expression of a relation by hand-crafted rules. In our work we attempt to solve this issue by applying machine learning techniques to learn the models of axioms, thus avoiding hand-crafted patterns on the lexicon or the syntactic structure of a sentence and instead implicitly learning probable language relation expressions.

4 Conclusion

In this work we addressed a novel problem of generating formal definitions from textual descriptions of biomedical concepts. Formal definition generation is a complex task. We approached it from a text mining perspective and split it into several consecutive steps. We were particularly focused on the non-taxonomic relation extraction as expressive relations contain the core information about the concepts to be defined. We implemented and evaluated relation extraction and relation classification steps, integrating state-of-the-art domain resources and external tools, i.e. semantic annotators, achieving high-performance and setting the scene for further research of the FDG problem. Formal definition generation pipeline can be used as a standalone tool for concept formalization, and it can also be integrated into ontology learning tools as a semi-automatic tool for the assistance to domain experts.

References

1. F. Baader, I. Horrocks and U. Sattler. *Description logics*. In Handbook on Ontologies, 2004.
2. G. Tsatsaronis, A. Petrova, M. Kissa, Y. Ma, F. Distel, F. Baader, M. Schroeder. *Learning Formal Definitions for Biomedical Concepts*. In OWLED, 2013.
3. O. Bodenreider. *The Unified Medical Language System (UMLS): integrating biomedical terminology*. Nucleic Acids Research, 32(Database issue): D267-D270, Jan 2004.
4. D. L. Rubin, O. Dameron, Y. Bashir, D. Grossman, P. Dev, and M. A. Musen. *Using ontologies linked with geometric models to reason about penetrating injuries*. Artificial Intelligence in Medicine, 37(3):167-176, 2006.
5. R. D. King, J. J. Rowland, W. Aubrey, M. Liakata, M. Markham, L. N. Soldatova, K. E. Whelan, A. Clare, M. Young, A. Sparkes, S. G. Oliver, and P. Pir. *The robot scientist Adam*. IEEE Computer, 42(8):46-54, 2009.
6. T. Wächter. 2010. *Semi-automated Ontology Generation for Biocuration and Semantic Search*. PhD thesis. Technische Universität Dresden, Germany.
7. A. R. Aronson. *MetaMap: Mapping Text to the UMLS Metathesaurus*. Bethesda, MD: NLM, NIH, DHHS (2006).
8. Unified Medical Language System, <http://www.nlm.nih.gov/research/umls/>
9. S. Schulze-Kremer, B. Smith, and A. Kumar. *Revising the UMLS semantic network*. Medinfo (2004): 1700-4.
10. J. Volker, P. Hitzler and P. Cimiano. *Acquisition of OWL DL axioms from lexical resources*. In ESWC, pages 670-685, 2007.
11. P. Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer-Verlag New York, Inc., 2006.
12. A. Petrova. *Learning Formal Definitions for Biomedical Concepts*. Master thesis. Technische Universität Dresden, Germany.