Text Onto Miner – A Semi Automated Ontology Building System

Piotr Gawrysiak[1], Grzegorz Protaziuk[1], Henryk Rybinski[1], and Alexandre Delteil[2]

[1] ICS, Warsaw University of Technology
[2] France Telecome R & D
{P.Gawrysiak,G.Protaziuk,H.Rybinski}@ii.pw.edu.pl,
alexandre.delteil@orange-ft.com

Abstract. This paper presents an overview of the results of the project undertaken by the Warsaw University of Technology Institute of Computer Science as a part of research agreement with France Telecom. The project goal was to create a set of tools – both software and methods, that could be used to speed up and improve a process of creating ontologies. In the course of the project a new ontology building methodology has been devised, new text mining algorithms optimized for extracting information useful for building an ontology from text corpora have been proposed and an universal text mining toolkit – TOM Platform – have been implemented.

Keywords: Natural language processing, ontologies, text mining.

1 Introduction

Ontologies have shown their importance in many application areas, such as intelligent knowledge base integration, information brokering, and natural-language processing, just to indicate few of them. Their importance is growing, as is growing on the Web the number of information repositories that need metadata enrichment and analysis. On the other hand however, their usage is still very limited by ontology engineering, which is very time-consuming and expensive. Therefore there is a growing need for automated – or at least semi-automated methods, that will be able to leverage the amount of information present in ever growing repositories of text data (e.g. obtainable via the Internet) in order to build useful ontology systems.

One can distinguish two main approaches in discovering semantic information from text corpora – knowledge-rich and knowledge-poor ones, according to the amount of knowledge they presuppose [7]. Knowledge-rich approaches require some sort of previously built semantic information, domain-dependent knowledge structures, semantic tagged training corpora, or semantic dictionaries, thesauri, ontologies, etc. (see e.g. [9,24]). In most cases, the reported methods refer to knowledge-rich methods, which require deep and specific knowledge "coded" into the algorithms, auxiliary dictionaries and/or thesauri, very much language and domain dependent. Although the knowledge rich methods may

A. An et al. (Eds.): ISMIS 2008, LNAI 4994, pp. 563–573, 2008.

bring better results than the knowledge poor ones, the requirement for deep and specific knowledge is the main limitation in using them. There is therefore a high demand for finding knowledge-poor algorithms that would give satisfactory results, especially for the cases of limited lexical resources. In this context, the most promising approach seems to be utilization of text mining techniques for discovering semantic information from text corpora in order to build or maintain ontologies (see e.g. [14]). To this date most approaches of this type have only looked at a very specific problem, e.g. how to learn the taxonomic part of ontologies, or how to find proper names. The most complete approach has been reported in the work performed by the group of AIFB (Karlsruhe University), e.g. [14,15]. Our approach follows this direction. So, as in [14] we attempt to cover the entire process of ontology building, and provide an advanced platform (named TOM), supporting the whole process. Additionally, we have incorporated to TOM a number of novel algorithms, focused mainly on enriching the ontologies lexical layer. TOM integrates text preprocessing algorithms with novel TM based algorithms [19,21,11,22], and the tools for merging partial results into the existing ontology.

In this paper we present an overview of this approach, and the set of tools. It is a semi-automated method that could help building ontologies thanks to the analysis and extraction of semantic information from large text corpora. The structure of this paper is as follows: Section 2 presents an overview of the proposed ontology building process, and Section 3 describes briefly new algorithms and methods that have been developed in order to support this process. Section 4 presents the structure of the TOM platform that has been developed specifically in order to verify experimentally the proposed algorithms and the methodology. Finally, Section 5 contains information concerning experimental results and concluding remarks.

2 Semi-automatic Ontology Building Process

In the literature many approaches to building ontologies have been introduced and discussed (see for example) [2,1,4,5,6,12,16,23]. In [8] and [17] authors present an opinion that the process of ontology building is not a rigorous engineering discipline. Nevertheless, tasks that are required in order to create ontology are quite well defined. According to e.g. [17] these include: (a) defining a domain and scope of an ontology, (b) creating a comprehensive list of concepts (classes) and their hierarchy, (c) defining relations between classes and (d) populating an ontology with instances of classes. Additionally, some auxiliary tasks, such as defining the properties of classes or preserving transitivity of some relations (e.g. taxonomy, part_of), and avoiding cycles, are usually required.

As shown in [20], there are already many publications referring to the research on automatic tools that support ontology building process in various phases. In many of above tasks (e.g. while determining relations between classes) some automatic tools can be used, to a higher or lesser extent. Some of these tasks cannot be even performed manually in a reasonable amount of time. This statement refers specifically to situations, where a huge amount of data should be

processed and/or analyzed. With the text mining methods we can provide a support in such cases, however the discovered knowledge always requires human decision and intervention, thus provided tools will be always semi-automatic.

Below we sketch a method of building a domain ontology from unstructured text documents with the text mining support provided by the TOM text mining platform. We propose an approach to building an ontology from a domain specific repository. We assume that neither an a priori taxonomy nor ontology is available. The approach consists of the following steps:

1. Text extraction and preprocessing
TOM provides a variety of tools for text preprocessing. In various text mining experiments there may be different needs for defining a text unit. We have therefore introduced an option for defining granularity of the text mining process. In particular, TOM allows viewing the whole corpus as a set of documents, paragraphs or sentences. For example, for experiments aiming at discovering compound terms [19], or synonyms [21] the granularity was set to the sentence level. For discovering homonyms and homograms [22] we have performed experiments with the granularity set at the paragraph level. The process can be defined as a pipeline (Fig. 1), and the results can be used in all the other phases.

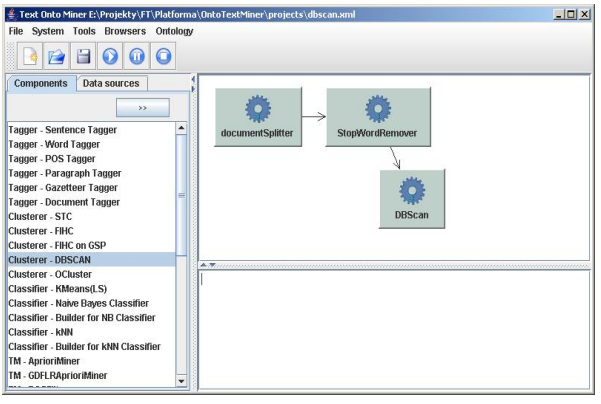

Fig. 1. Designing a preprocessing pipeline

2. Determining list of terms & purifying document representations
In this step two kinds of terms are determined: (a) words specific for a domain of interest, and (b) compound terms, including compound proper nouns (see [19] for details). For building the list (a) usually one should extract a dictionary from the given repository and subtract from it the dictionary received from a reference repository (containing a common sense texts). Next, sequences of words representing multiword terms are replaced by compound terms in the text.

3. Building a hierarchy (taxonomy)
The candidates for the hierarchy building are discovered with FIHC, and *apriori* (see [11] and [21] for details respectively). Based on it, the user selects proper candidates. It is difficult to automatically detect the hierarchy for compound proper nouns, because such terms occur in texts rather infrequently, and they are disseminated among many clusters. It is therefore reasonable in FIHC to ignore compound (proper) nouns. On the other hand, synonymy discovering

procedure [21] finds *inter alia* categories of terms. In such cases a taxonomy level can be created or adjusted manually, when needed. Then hierarchies are merged. The step ends with an ontology skeleton.

4. Discovering terms meanings
By discovering terms meanings we (1) identify the pairs of terms with close meanings, and homonyms (or in general, various meanings of some terms). To this end, we use the methods sketched in Section 3. As the method described discovers also pairs of the type (broader, narrower), part_of, or belonging to the same category, some results can be used to enrich the taxonomy obtained in the steps above. Discovered synonyms and homonyms can be used for enriching the lexical part of the ontology by relating the literals to the appropriate concepts. In the case of homonyms we also keep within the ontology various meanings of the words. The information can be then used by the user for enriching the dictionary layer and its connections to the concept layer of the ontology.

5. Discovering association relations & enriching the ontology skeleton
The process of discovering association relations is done by applying the method described in Section 3. Discovered relations can be then used by the user for building the concept layer of the ontology. In this case each term included in the discovered relations is treated as an instance of a certain class unless it is directly stated that the word is a name of a class. The OWL standard requires that each instance must have assigned a class but it may happen that for some instances appropriate classes are unknown.

It is worth noting that steps 2 and 5 are performed with the same T-GSP algorithm (though with different parameters), whereas step 4 (both homonymy and close meaningterm discovery) is performed with *apriori* [1].

3 New Text Analysis Methods

During the course of the TOM project several new text analysis methods have been developed. The methods mainly refer to discovering various relations between words and their properties relevant in the ontology development and maintenance process. In particular, the following algorithms have been elaborated and tested, and then incorporated into TOM:

1. discovering hierarchies with FIHC algorithm [6]; the algorithm is used for finding taxonomies from the set of documents; a number of improvements have been introduced to the algorithm in comparison to the original, making the algorithm much faster;
2. *apriori* based algorithm for discovering close meaning terms (synonyms, antonyms, BT/NT related terms, category/instances related terms, etc.);
3. *apriori* based algorithm for discovering homonyms and homograms;
4. T-GSP algorithm for discovering grammatically filtered frequent patterns; the algorithm can be used for discovering (proper) composed nouns, as well as association relations, depending on the grammar rules defined;

5. new (knowledge rich) methods for discovering association relations;
6. a number of rules have been elaborated in order to support the process of merging ontologies.

Some methods have been described in detail in [21,19,11,22]. A complete description is provided in [18]. Below we briefly present only short descriptions of the selected algorithms.

Discovering hierarchies based on the FIHC

Processing text corpora with the FIHC algorithm [6] generates hierarchical clustering tree. As described in [6], labels of this tree tend to form a potentially interesting taxonomy. Resulted tree comes from the frequent itemsets tree, which has taxonomic character by itself, e.g. *Warsaw University of Technology* is obviously in is a relation with University and the former itemset is a superset of the latter one. In addition, the obtained FIHC tree has the following advantages over normal frequent sets trees:

- it is not full. In the case of frequent sets, if an itemset is frequent, then all its subsets are also frequent, and all of them exist in the tree. FIHC takes only the most significant subsets.
- FIHC can merge similar siblings, so it might happen that a phrase has a parent which is not its subset.

In [11] our group have proved that the original algorithm can easily be modified to operate on closed sets and can produce exactly the same results as by means of frequent itemsets. The number of frequent closed sets is usually much lower than the number of frequent itemsets giving significant gain in time of execution and memory requirements. Moreover, we made some modifications to the algorithm which strengthen the ability of discovering taxonomies. We experimented with many modifications. The ones listed below significantly improve the taxonomy discovery:

- use of POS tagging for pattern matching & pre-filtering. It allows restricting search only for relations of a given form, e.g. only noun-noun relations. We can define any pattern which can be expressed by means of regular expressions.
- allowing obvious pruning only. Normally, the resultant tree is vastly pruned in the last phases of the FIHC algorithm. This option allows performing only obvious pruning. It means that parent-child merging is allowed only in case of one-child nodes. This option was introduced to retrieve more semantic relations. Originally FIHC was meant to produce labeled clustering tree and it was reasonable to have a small number of meaningful groups. We are not interested in clusters, instead, we want to have a rich description of semantic relations.
- pulling up labels. While performing the pruning, original FIHC leaves a parent label that is the shorter one. In our case, we do not look for cluster labels, but we need semantic relations, which are more likely hidden in longer labels. When the option of pulling labels is on, the label of the child is kept instead of the parent's one during the pruning phase.

- operating on sequential patterns instead of itemsets, which keeps words ordering, and makes results more readable and potentially more accurate. We replaced the phase of itemsets discovery with TGSP.

Discovering synonyms [21]
In our method we assume that: *The synonyms do not appear together in a sentence, but they appear quite frequently in similar contexts.* Hence, synonyms are often used with the same words. Having the pairs of terms that do not co-occur, we apply the similarity measures CSIM and ASIM [21]. The approach for generating pairs of terms that are likely to be synonyms consists of the following steps:

1. Text corpus is tagged with parts of speech, and some cleaning is done;
2. Text data are converted into transactional database, where sentences are treated as transactions (by means of [1];
3. The *apriori* based algorithm for finding frequent itemsets is executed;
4. For every frequent term being in the field of interest, frequent itemsets containing that word are found (we call the set of itemsets context of a word);
5. Finally synonymy measure is computed for every pair of words with use of their contexts. Based on this, a decision is taken whether the pair can candidate for synonymy or not.

Discovering homonyms
Distinct meanings of homonyms are indicated by various distinct contexts in which they appear frequently. This assumption is based on the distributional hypothesis [10], where the underlying idea is that "*a word is characterized by the company it keeps*". The rule is very intuitive, and therefore is applied to the proposed approaches. The problem is, however, how the notion of a context is defined. For example, it can be understood as a set of words surrounding a target word frequently enough in documents, paragraphs, or sentences. In our approach, context is evaluated as below.

Let dictionary $D = \{t_1, t_2, \ldots, t_m\}$ be a set of distinct words, called *terms*. In general, any set of terms is called a termset. The set $\mathcal{P}$ is a set of *paragraphs*, where each paragraph P is a set of terms such that $P \subseteq \mathcal{P}$.

Statistical significance of a termset X is called *support* and is denoted by $sup(X)$. $sup(X)$ is defined as the number (or percentage) of paragraphs in $\mathcal{P}$ that contain X. Clearly, the supports of termsets that are supersets of termset X are not greater than $sup(X)$.

A termset is called *frequent* if it occurs in more than ε paragraphs in $\mathcal{P}$, where ε is a user-defined support threshold. In the sequel, we will be interested in maximal frequent termsets, which we will denote by MF and define as the set of all maximal (in the sense of inclusion) termsets that are frequent.

Let x be a term. By $MF(x)$ we denote all maximal frequent termsets containing x. $MF(x)$ will be used for determining *atomic contexts* for x. A termset $X, x \notin X$, is defined as an *atomic context* of term x if $\{x\} \cup X$ is an element of $MF(x)$. The set of all atomic contexts of x will be denoted by $AC(x)$:

$$AC(x) = \{X \backslash \{x\} \mid X \in MF(x)\}.$$

Clearly, for each two termsets Y, Z in $AC(x)$, Y differs from Z by at least one term and vice versa. In spite of this, Y and Z may indicate the same meaning of x in reality. Let y be a term in $Y \backslash Z$ and z be a term in $Z \backslash Y$ and $\{xyz\}$ be a termset the support of which is significantly less than the supports of Y and Z. This may suggest that Y and Z probably represent different meanings of x. Otherwise, Y and Z are likely to represent the same meaning of x. Please, note that $\{xyz\}$ plays a role of a *potential discriminant* for pairs of atomic contexts. The set of all potential discriminants for Y and Z in $AC(x)$, denoted by $\mathcal{D}(x, Y, Z)$ is:

$$\mathcal{D}(x, Y, Z) = \{\{xyz\} \mid y \in Y \backslash Z \wedge z \in Z \backslash Y\}.$$

Among the potential discriminants, those which are relatively infrequent are called *proper discriminants*. Formally, the set of *all proper discriminants* for Y and Z in $AC(x)$ will be denoted by $\mathcal{PD}(x, Y, Z)$, and defined as follows:

$$\mathcal{PD}(x, Y, Z) = \{X \in \mathcal{D}(x, Y, Z) \mid relSup(x, X, Y, Z) \leq \delta\}, \text{ where}$$
$$relSup(x, X, Y, Z) = sup(X)/min(sup(xY), sup(xZ)), \text{ and}$$
$$\delta \text{ is a user-defined threshold.}$$

In the sequel, $relSup(x, X, Y, Z)$ is called a *relative support of discriminant* X for term x with respect to atomic contexts Y and Z.

Our proposal of determining the groups of contexts representing separate meanings of x is based on the introduced notion of proper discriminants for pairs of atomic contexts.

Atomic contexts Y and Z in $AC(x)$ are called *discriminable* if there is at least one proper discriminant in $\mathcal{PD}(x, Y, Z)$. Otherwise, Y and Z are called *indiscriminable*.

A *sense-discriminant context* $\mathcal{SDC}(x, X)$ of x for termset X in $AC(x)$ is defined as the family of the termsets in $AC(x)$ that are indiscriminable with X:

$$\mathcal{SDC}(x, X) = \{Y \in AC(x) \mid \mathcal{PD}(x, X, Y) = \emptyset\}.$$

Clearly, $X \in \mathcal{SDC}(x, X)$. Please, note that sense-discriminant contexts of x for Y and Z, where $Y \neq Z$, may overlap, and in particular, may be equal.

The family of all distinct sense-discriminant contexts, denoted by $\mathcal{FSDC}(x)$ is:

$$\mathcal{FSDC}(x) = \{\mathcal{SDC}(x, X) \mid X \in AC(x)\}.$$

Please, note that $\mid \mathcal{FSDC}(x) \mid \leq \mid AC(x) \mid$.

A given term x is defined as a *homonym candidate* if the cardinality of $\mathcal{FSDC}(x)$ is greater than 1. Final decision on homonymy is given to the user. Let us also note that the more overlapping are distinct sense-discriminant contexts, the more difficult is reusing the contexts for the meaning recognition in the mining procedures.

Merging ontologies

Having loaded an ontology, one can merge it with detected proposals. In such a process, a number of conflicts may occur, e.g. the type of an attribute is different in both ontologies. The conflicts can be resolved either automatically or manually, though, some can be resolved only manually, e.g., if we have an attribute

definition of *integer* type in the base ontology and an attribute definition of *string* type in the other one, we will have a conflict that cannot be resolved automatically. The users may set the way of resolving conflicts with conflict resolving setup window, which is presented on the picture below. The available options associated with a particular kind of conflicts are organized in the hierarchy where leafs represent single conflicts.

4 The TOM Platform

The TOM platform is thought as a universal environment that allows easy experimentations with various text mining algorithms for ontology building. The system is highly modular (based on plug-in architecture), and highly portable as Java has been used as the implementation language. At the top level it consists of the following subsystems:

Text mining subsystem. The text mining subsystem enables a user to specify both data source of an experiment and a pipeline of a text mining process (Fig. 1). Such a pipeline may consists of several steps for text processing, e.g. generation of the bag of words representation of documents, splitting the text into sentences, etc. In addition some text mining algorithms may be used as a step within a pipeline, for example clustering of documents, discovering frequent multi word terms, etc.

Analysis support subsystem. The analysis support subsystem is dedicated for working with the results obtained from text mining plan-and-experiment subsystem. For each type of results (clusters, sequences, candidate homonyms, etc.) the dedicated tools supporting basic analysis are available. Also the viewer of the *ftdoc* type documents is provided.

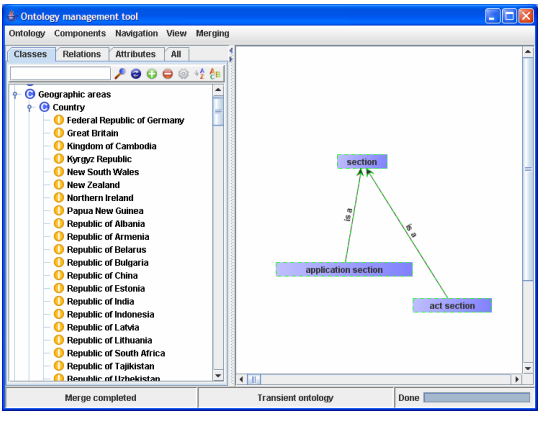

Fig. 2. Ontology managament subsystem

Ontology subsystem
The ontology subsystem is thought as a tool with a convenient graphical interface for working on ontologies, for example enriching ontologies, based on results obtained from text-mining experiments. It enables users to do various operations: such as browsing, annotating or validating ontologies. The subsystem also includes a tool by which a user may generate the owl file with proposals of new entries to ontologies. Such proposals are generated based on results obtained by using text-mining algorithm implemented in TOM.

Data storing subsystem. The data storing subsystem provides functionality concerning saving and searching for all text data used in the TOM platform. An integral part of this subsystem is an indexer (in TOM the Lucene [13] index is used). The indexer helps one to create fast-searchable database of text documents backed up with inverted index and vector document representation.

5 Project Results and Conclusions

For the experimental evaluation of the algorithms a FAO[1] document repository from the FAOLEX system was used. The full repository consists of more than 20000 national legislation documents concerning legal issues for food and agriculture. The repository is provided in various languages. We have selected 5658 documents written in English documents. Of it, we were able to extract 546.617 paragraphs and then 1.296.929 sentences. The experiments conducted on this corpus showed that especially the methods proposed by us for automatic homonymy and synonymy identification were giving very robust results (in many cases discovering information unknown previously to us, such as semantic relationship between names of various species and species groups). The hierarchy building algorithms do not eliminate the need for human intervention, they are very useful in the ontology building process, and definitely speed up the overall engineering process in ontology building – especially helpful was the ability to quickly generate a rough ontology skeleton that could be further improved by domain experts. The illustrations below present a small fragment of ontology that has been created with TOM by experiments run on the Faolex repository mentioned above.

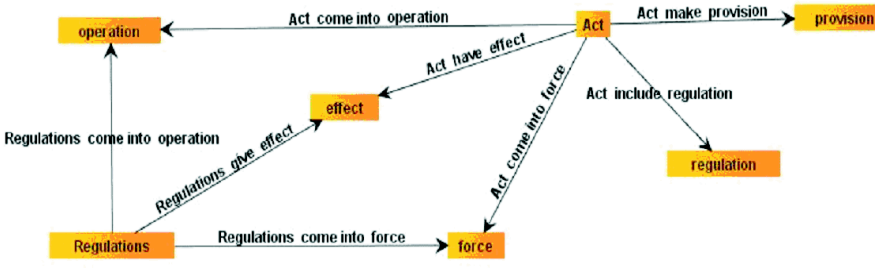

Fig. 3. A subset of ontology created from FAOEX repository with TOM

The research briefly described in this paper is far from being complete. While the initial project, that has been commission by France Telecom, has been completed successfully (i.e. the TOM platform can be readily used in order to speed up ontology building and to improve the quality of resulting ontologies) we will be extending the TOM platform system, with special emphasis on ontology merging and introduction of more knowledge-rich methods into the platform.

[1] FAO is a UN organization (Food and Agriculture Organization).

Acknowledgments. The work described in this paper results from the project funded by France Telecom.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. of the 20th Int'l. Conf. on VLDB, Santiago, Chile, Morgan Kaufmann, San Francisco (1994)
2. Ahonen-Myka, H.: Finding all frequent maximal sequences in text. In: Mladenic, D., Grobelnik, M. (eds.) Proc. of the 16th Int. Con. on Machine Learning ICML 1999 Workshop on Machine Learning in Text Data Analysis, pp. 11–17 (1999)
3. Beil, F., Ester, M., Xu, X.: Frequent term-based text clustering. In: KDD 2002 (2002)
4. Byrd, R., Ravin, Y.: Identifying and extracting relations from text. In: NLDB 1999 - 4th Int. Con. on Applications of Natural Language to Information Systems (1999)
5. Faure, D., Nedellec, C.: A corpus-based conceptual clustering method for verb frames and ontology acquisition. In: LREC Workshop on Adapting Lexical and Corpus Resources to Sublanguages and Applications, Granada, Spain (1998)
6. Fung, B.C.M., Wan, K., Ester, M.: Hierarchical document clustering Using Frequent Item-sets. In: SDM 2003 (2003)
7. Grefenstette, G.: Evaluation Techniques for Automatic Semantic Extraction: Comparing Syntatic and Window Based Approaches. In: Boguraev, B., Pustejovsky, J. (eds.) Corpus processing for Lexical Acquisition, pp. 205–216. MIT Press, Cambridge (1995)
8. Guarino, N., Welty, C.: Evaluating ontological decisions with Ontoclean. Comm. of ACM 45(2) (2002)
9. Hamon, T., Nazarenko, A., Gros, C.: A step towards the detection of semantic variants of terms in technical documents. In: Proc. 36th Ann. Meeting of ACL (1998)
10. Harris, Z.: Distributional structure. Word 10(23), 146–162 (1954)
11. Skonieczny, K.M.: Hierarchical document clustering using frequent closed sets. In. Proc. IIPWM (2006)
12. Lame, G.: Using text analysis techniques to identify legal ontologie's components. In: ICAIL 2003, Workshop on Legal Ontologies & Web Based Legal Inf. Manag. (2003)
13. Lucene home page, `http://www.apache.org/lucene`
14. Maedche, A., Staab, S.: Ontology Learning, Handbook on Ontologies. Springer Series on Handbooks in Information Systems. Springer, Heidelberg (2003)
15. Maedche, A., Staab, S.: Mining Ontologies from Text. In: Dieng, R., Corby, O. (eds.) EKAW 2000. LNCS (LNAI), vol. 1937, pp. 189–202. Springer, Heidelberg (2000)
16. Morin, E.: Automatic acquisition of semantic relations between terms from technical corpora. In: Proc. 5th Int'l. Congress on TKE (1999)
17. Noy, F.N., McGuinness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Techn. Rep. SMI-2001-0880
18. Protaziuk, G., et al.: TOM Platform Reference Manual, Techn. Rep., WUT (2006)
19. Protaziuk, G., et al.: Discovering Compound and Proper Nouns. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, Springer, Heidelberg (2007)

20. Protaziuk, G., et al.: State of The Art on Ontology and Vocabulary Building & Maintenance Research And Applications, Techn. Rep., WUT (2006)
21. Rybinski, H., et al.: Discovering Synonyms based on Frequent Termsets. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, Springer, Heidelberg (2007)
22. Rybinski, H., et al.: Discovering Word Meanings Based on Frequent Termsets. In: MCD Workshop, PKDD, Warsaw (2007)
23. Velardi, P., Fabriani, P., Missikoff, M.: Using text processing techniques to automatically enrich a domain ontology. In: Proc. Int'l. Conf. on FOIS (2001)
24. Wu, H., Zhou, M.: Optimizing Synonym Extraction Using Monolingual and Bilingual Resources. In: Ann. Meeting ACL, Proc. 2^{nd} Int'l Workshop on Paraphrasing, vol. 16, pp. 72–79 (2003)