Weakly Supervised Approaches for Ontology Population

Hristo Tanev Tanev
ITC-irst
38050, Povo
Trento, Italy
htanev@yahoo.co.uk

Bernardo Magnini
ITC-irst
38050, Povo
Trento, Italy
magnini@itc.it

Abstract

We present a weakly supervised approach to automatic Ontology Population from text and compare it with other two unsupervised approaches. In our experiments we populate a part of our ontology of Named Entities. We considered two high level categories - *geographical locations* and *person names* and ten sub-classes for each category. For each sub-class, from a list of training examples and a syntactically parsed corpus, we automatically learn a *syntactic model* - a set of weighted *syntactic features*, i.e. words which typically co-occur in certain syntactic positions with the members of that class. The model is then used to classify the unknown Named Entities in the test set. The method is weakly supervised, since no annotated corpus is used in the learning process. We achieved promising results, i.e. 65% accuracy, outperforming significantly previous unsupervised approaches.

1 Introduction

Automatic Ontology Population (OP) from texts has recently emerged as a new field of application for knowledge acquisition techniques (see, among others, (Buitelaar et al., 2005)). Although there is no a univocally accepted definition for the OP task, a useful approximation has been suggested (Bontcheva and Cunningham, 2003) as Ontology Driven Information Extraction, where, in place of a template to be filled, the goal of the task is the extraction and classification of instances of concepts and relations defined in a Ontology. The task has been approached in a variety of similar perspectives, including term clustering (e.g. (Lin, 1998a) and (Almuhareb and Poesio, 2004)) and term categorization (e.g. (Avancini et al., 2003)).

A rather different task is Ontology Learning (OL), where new concepts and relations are supposed to be acquired, with the consequence of changing the definition of the Ontology itself (see, for instance, (Velardi et al., 2005)).

In this paper OP is defined in the following scenario. Given a set of terms $T = t_1, t_2, ..., t_n$, a document collection D, where terms in T are supposed to appear, and a set of predefined classes $C = c_1, c_2, ..., c_m$ denoting concepts in an Ontology, each term t_i has to be assigned to the proper class in C. For the purposes of the experiments presented in this paper we assume that (i) classes in C are mutually disjoint and (ii) each term is assigned to just one class.

As we have defined it, OP shows a strong similarity with Named Entity Recognition and Classification (NERC). However, a major difference is that in NERC each occurrences of a recognized term has to be classified separately, while in OP it is the term, independently of the context in which it appears, that has to be classified.

While Information Extraction, and NERC in particular, have been addressed prevalently by means of supervised approaches, Ontology Population is typically attacked in an unsupervised way. As many authors have pointed out (e.g. (Cimiano and Völker, 2005)), the main motivation is the fact that in OP the set of classes is usually larger and more fine grained than in NERC (where the typical set includes Person, Location, Organization, GPE, and a Miscellanea class for all other kind of entities). In addition, by definition, the set of classes in C changes as a new ontology is considered, making the creation of annotated data almost impossible practically.

According with the demand for weakly supervised approaches to OP, we propose a method, called $Class - Example$, which learns a classification model from a set of classified terms, exploiting lexico-syntactic features. Unlike most of the approaches which consider pair wise similarity between terms ((Cimiano and Völker, 2005); (Lin, 1998a)), the Class-Example method considers the similarity between a term t_i and a set of training examples which represent a certain class. This results in a great number of class features and opens the possibility to exploit more statistical data, such as the frequency of appearance of a class feature in different training terms.

In order to show the effectiveness of the Class-Example approach, it has been compared against two different approaches: (i) a *Class-Pattern* unsupervised approach, in the style of (Hearst, 1998); (ii) an unsupervised approach that considers the word of the class as a pivot word for acquiring relevant contexts for the class (we refer to this method as $Class-Word$). Results of the comparison show that the Class-Example method outperforms significantly the other two methods, making it appealing even considering the need of supervision.

Although the Class-Example method we propose is applicable in general, in this paper we show its usefulness when applied to terms denoting Named Entities. The motivation behind this choice is the practical value of Named Entity classifications, as, for instance, in applications such as Questions Answering and Information Extraction. Moreover, some Named Entity classes, including names of writers, athletes and organizations, dynamically change over the time, which makes it impossible to capture them in a static Ontology.

The rest of the paper is structured as follows. Section 2 describes the state-of-the-art methods in Ontology Population. Section 3 presents the three approaches to the task we have compared. Section 4 introduces Syntactic Network, a formalism used for the representation of syntactic information and exploited in both the Class-Word and the Class-Example approaches. Section 5 reports on the experimental settings, results obtained, and discusses the three approaches. Section 6 concludes the paper and suggests directions for future work.

2 Related Work

There are two main paradigms distinguishing Ontology Population approaches. In the first one Ontology Population is performed using patterns (Hearst, 1998) or relying on the structure of terms (Velardi et al., 2005). In the second paradigm the task is addressed using contextual features (Cimiano and Völker, 2005).

Pattern-based approaches search for phrases which explicitly show that there is an "is-a" relation between two words, e.g. "the ant is an insect" or "ants and other insects". However, such phrases do not appear frequently in a text corpus. For this reason, some approaches use the Web (Schlobach et al., 2004). (Velardi et al., 2005) experimented several head-matching heuristics according to which if a $term_1$ is in the head of $term_2$, then there is an "is-a" relation between them: For example "Christmas tree" is a kind of "tree".

Context feature approaches use a corpus to extract features from the context in which a semantic class tends to appear. Contextual features may be superficial (Fleischman and Hovy, 2002) or syntactic (Lin, 1998a), (Almuhareb and Poesio, 2004). Comparative evaluation in (Cimiano and Völker, 2005) shows that syntactic features lead to better performance. Feature weights can be calculated either by Machine Learning algorithms (Fleischman and Hovy, 2002) or by statistical measures, like Point Wise Mutual Information or the Jaccard coefficient (Lin, 1998a).

A hybrid approach using both pattern-based, term structure, and contextual feature methods is presented in (Cimiano et al., 2005).

State-of-the-art approaches may be divided in two classes, according to different use of training data: Unsupervised approaches (see (Cimiano et al., 2005) for details) and supervised approaches which use manually tagged training data, e.g. (Fleischman and Hovy, 2002). While state-of-the-art unsupervised methods have low performance, supervised approaches reach higher accuracy, but require the manual construction of a training set, which impedes them from large scale applications.

3 Weakly supervised approaches for Ontology Population

In this Section we present three Ontology Population approaches. Two of them are unsupervised:

(i) a pattern-based approach described in (Hearst, 1998), which we refer to as *Class-Pattern* and (ii) a feature similarity method reported in (Cimiano and Völker, 2005) to which we will refer as *Class-Word*. Finally, we describe a new weakly supervised approach for ontology population which accepts as a training data lists of instances for each class under consideration. This method we call *Class-Example*.

3.1 Class-Pattern approach

This approach was described first in (Hearst, 1998). The main idea is that if a term t belongs to a class c, then in a text corpus we may expect the occurrence of phrases like *such c as t,...*. In our experiments for ontology population we used the patterns described in the Hearst's paper plus the pattern *t is (a | the) c*:

1. t is (a | the) c
2. such c as t
3. such c as (NP,)*, (and | or) t
4. t (,NP)* (and | or) other c
5. c, (especially | including) (NP,)* t

For each instance from the test set t and for each concept c we instantiated the patterns and searched with them in the corpus. If a pattern which is instantiated with a concept c and a term t appears in the corpus, then we assume the t belongs to c. For example, if the term to be classified is "Etna" and the concept is "mountain", one of the instantiated patterns will be "mountains such as Etna"; if this pattern is found in the text, then "Etna" is considered to be a "mountain". If the algorithm assigns a term to several categories, we choose the one which co-occurs most often with the term.

3.2 Class-Word approach

(Cimiano and Völker, 2005) describes an unsupervised approach for ontology population based on vector-feature similarity between each concept c and a term to be classified t. For example, in order to conclude how much "Etna" is an appropriate instance of the class "mountain", this method finds the feature-vector similarity between the word "Etna" and the word "mountain". Each instance from the test set T is assigned to one of the classes in the set C. Features are collected from $Corpus$ and the classification algorithm on figure 1 is applied. The problem with this approach is that the context distribution of a name (e.g. "Etna") is sometimes different than the context distribution of the class word (e.g. "mountain"). Moreover, a single word provides a limited quantity of contextual data.

```
classify($T$, $C$, $Corpus$)
    foreach($t$ in $T$) do{
        $v_t = getContextVector(t, Corpus);$}
    foreach($c$ in $C$) do{
        $v_c = getContextVector(c, Corpus);$}
    foreach($t$ in $T$) do{
        $classes[t] = argmax_{c \in C} sim(v_t, v_c);$}
    return $classes[]$;
end classify
```

Figure 1: Unsupervised algorithm for Ontology Population.

In this algorithm the context vectors v_t and v_c are feature vectors whose elements represent weighted context features from $Corpus$ of the term t (e.g. "Etna") or the concept word c (e.g. "mountain"). Cimiano and Völker evaluate different context features and prove that syntactic features work best. Therefore, in our experimental settings we considered only such features extracted from a corpus parsed with a dependency parser. Unlike the original approach which relies on pseudo-syntactic features, we used features extracted from dependency parse trees. Moreover, we used virtually all the words connected syntactically to a term, not only the modifiers. A syntactic feature is a pair: *(word, syntactic relation)* (Lin, 1998a). We use two feature types: *First order features*, which are directly connected to the training or test examples in the dependency parse trees of $Corpus$; *second order features*, which are connected to the training or test instances indirectly by skipping one word (the verb) in the dependency tree. As an example, let's consider two sentences: "Edison invented the phonograph" and "Edison created the phonograph". If "Edison" is a name to be classified, then two first order features of this name exist - ("invent", *subject-of*) and ("create", *subject-of*). One second order feature can be extracted - ("phonograph", *object-of+subject*); it co-occurs two times with the word "Edison". In our experiments second order features are considered only those words which are governed by the same verb whose subject is the name which is a training or test instance (in this example "Edison").

3.3 Weakly Supervised Class-Example Approach

The approach we put forward here uses the same processing stages as the one presented in Figure 1 and relies on syntactic features extracted from a corpus. However, the Class-Example algorithm receives as an additional input parameter the sets of training examples for each class $c \in C$. These training sets are simple lists of instances (i.e. terms denoting Named Entities), without context, and can be acquired automatically or semi-automatically from an existing ontology or gazetteer. To facilitate their acquisition, the Class-Example approach imposes no restrictions to the training examples - they can be ambiguous and have different frequencies. However, they have to appear in $Corpus$ (in our experimental settings - at least twice). For example, for the class "mountain" training examples are: "Everest", "Mauna Loa", etc.

The algorithm learns from each training set $Train(c)$ a single feature vector v_c, called the *syntactic model* of the class. Therefore, in our algorithm, the statement
$v_c = getContextVector(c, Corpus)$
in Figure 1 is substituted with
$v_c = getSyntacticModel(Train(c), Corpus)$.

For each class c, a set of syntactic features $F(c)$ are collected by finding the union of the features extracted from each occurrence in the corpus of each training instance in $Train(c)$. Next, the feature vector v_c is constructed: If a feature is not present in $F(c)$, then its corresponding coordinate in v_c has value 0; otherwise, it has a value equal to the feature weight.

The weight of a feature $f_c \in F(c)$ is calculated in three steps:

1. First, the co-occurrence of f_c with the training set is calculated:

$$weight_1(f_c) = \sum_{t \in Train(c)} \alpha.log(\frac{P(f_c, t)}{P(f_c).P(t)})$$

   where $P(f_c, t)$ is the probability that feature f_c co-occurs with t, $P(f_c)$ and $P(t)$ are the probabilities that f_c and t appear in the corpus, $\alpha = 14$ for syntactic features with lexical element noun and $\alpha = 1$ for all the other syntactic features. The α parameter reflects the linguistic intuition that nouns are more informative than verbs and adjectives which in most cases represent generic predicates. The values of α were automatically learned from the training data.

2. We normalize the feature weights, since we observed that they vary a lot between different classes: for each class c we find the feature with maximal weight and denote its weight with $mxW(c)$,

$$mxW(c) = max_{f_c \in F(c)} weight_1(f_c)$$

   Next, the weight of each feature $f_c \in F(c)$ is normalized by dividing it with $mxW(c)$:

$$weight_N(f_c) = \frac{weight_1(f_c)}{mxW(c)}$$

3. To obtain the final weight of f_c, we divide $weight_N(f_c)$ by the number of classes in which this feature appears. This is motivated by the intuition that a feature which appears in the syntactic models of many classes is not a good class predictor.

$$weight(f_c) = \frac{weight_N(f_c)}{|Classes(f_c)|}$$

   where $|Classes(f_c)|$ is the number of classes for which f_c is present in the syntactic model.

As shown in Figure 1, the classification uses a similarity function $sim(v_t, v_c)$ whose arguments are the feature vector of a term v_t and the feature vector v_c for a class c. We defined the similarity function as the dot product of the two feature vectors: $sim(v_t, v_c) = v_c.v_t$. Vectors v_t are binary (i.e. the feature value is 1 if the feature is present and, 0-otherwise), while the features in the syntactic model vectors v_c receive weights according to the approach described in this Section.

4 Representing Syntactic Information

Since both the Class-Word and the Class-Example methods work with syntactic features, the main source of information is a syntactically parsed corpus. We parsed about half a gigabyte of a news corpus with MiniPar (Lin, 1998b). It is a statistically based dependency parser which is reported to reach 89% precision and 82% recall on press reportage texts. MiniPar generates syntactic dependency structures - directed labeled graphs whose

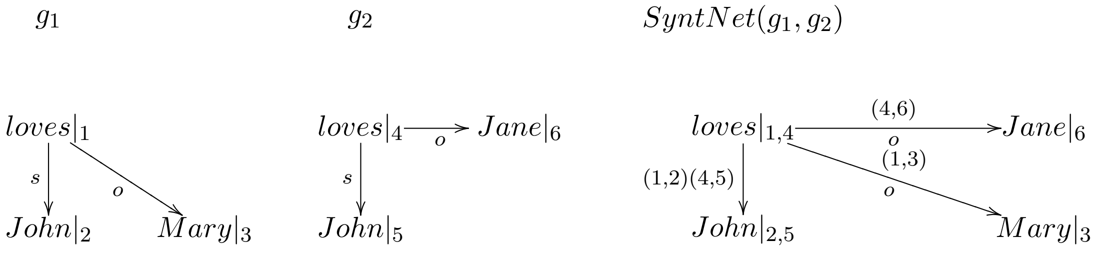

Figure 2: Two syntactic graphs and their Syntactic Network.

vertices represent words and the edges between them represent syntactic relations like *subject*, *object*, *modifier*, etc. Examples for two dependency structures - g_1 and g_2, are shown in Figure 2: They represent the sentences "John loves Mary" and "John loves Jane"; labels s and o on their edges stand for $subject$ and $object$ respectively. The syntactic structures generated by MiniPar are dendroid (tree-like), but still cycles appear in some cases.

In order to extract information from the parsed corpus, we had to choose a model for representing dependency trees which allows to search efficiently for syntactic structures and to calculate their frequencies. Building a classic index at word level was not an option, since we have to search for syntactic structures, not words. On the other hand, indexing syntactic relations (i.e. word pair and the relation between the words) would be useful, but still does not resolve the problem, since in many cases we search for more complex structures than a relation between two words: for example, when we have to find which words are syntactically related to a Named Entity composed by two words, we have to search for syntactic structures which consists of three vertices and two edges.

In order to trace efficiently more complex structures in the corpus, we put forward a model for representation of a set of labeled graphs, called *Syntactic Network* (*SyntNet* for short). The model is inspired by a model presented earlier in (Szpektor et al., 2004), however our model allows more efficient construction of the representation. The scope of SyntNet is to represent a set of labeled graphs through one aggregate structure in which the isomorphic sub-structures overlap. When SyntNet represents a syntactically parsed text corpus, its vertices are labeled with words from the text while edges represent syntactic relations from the corpus and are labeled accordingly.

An example is shown in Figure 2, where two syntactic graphs, g_1 and g_2, are merged into one aggregate representation $SyntNet(g_1, g_2)$. Vertices labeled with equal words in g_1 and g_2 are merged into one generalizing vertex in $SyntNet(g_1, g_2)$. For example, the vertices with label $John$ in g_1 and g_2 are merged into one vertex $John$ in $SyntNet(g_1, g_2)$.

Edges are merged in a similar way: $(loves, John) \in g_1$ and $(loves, John) \in g_2$ are represented through one edge $(loves, John)$ in $SyntNet(g_1, g_2)$.

Each vertex in g_1 and g_2 is labeled additionally with a numerical index which is unique for the graph set. Numbers on vertices in $SyntNet(g_1, g_2)$ show which vertices from g_1 and g_2 are merged in the corresponding SyntNet vertices. For example, vertex $loves \in SyntNet(g_1, g_2)$ has a set $\{1, 4\}$ which means that vertices 1 and 4 are merged in it. In a similar way the edge $(loves, John) \in SyntNet(g_1, g_2)$ is labeled with two pairs of indices $(4, 5)$ and $(1, 2)$, which shows that it represents two edges: the edge between vertices 4 and 5 and the edge between 1 and 2.

Two properties of SyntNet are important: first isomorphic sub-structures from all the graphs represented via a SyntNet are mapped into one structure. This allows us to easily find all the occurrences of multiword terms and named entities. Second, using the numerical indices on vertices and edges, we can efficiently calculate which structures are connected syntactically to the training and test terms. As an example, let's try to calculate in which constructions the word "Mary" appears considering $SyntNet$ in Figure 2. First, in SyntNet we can directly observe that there is the relation *loves* → *Mary* labeled with the pair $1 \rightarrow 3$ - therefore this relation appears once in the corpus. Next, tracing the numerical indices on the vertices and edges we can find a path from "Mary" to "John" through "loves". The path passes through the following numerical indices: $3 \leftarrow 1 \rightarrow 2$: this means that there is one appearance of the structure

"John loves Mary" in the corpus, spanning through vertices $1, 2$, and 3. Such a path through the numerical indices cannot be found between "Mary" and "Jane" which means that they do not appear in the same syntactic construction in the corpus.

SyntNet is built incrementally in a straightforward manner: Each new vertex or edge added to the network is merged with the identical vertex or edge, if such already exists in SyntNet. Otherwise, a new vertex or edge is added to the network. The time necessary for building a SyntNet is proportional to the number of the vertices and the edges in the represented graphs (and does not otherwise depend on their complexity).

The efficiency of the SyntNet model when representing and searching for labeled structures makes it very appropriate for the representation of a syntactically parsed corpus. We used the properties of SyntNet in order to trace efficiently the occurrences of Named Entities in the parsed corpus, to calculate their frequencies, to find the syntactic features which co-occur with these Named Entities, as well as the frequencies of these co-occurrences. Moreover, the SyntNet model allowed us to extract more complex, second order syntactic features which are connected indirectly to the terms in the training and the test set.

5 Experimental settings and results

We have evaluated all the three approaches described in Section 3. The same evaluation settings were used for the three experiments. The source of features was a news corpus of about half a gigabyte. The corpus was parsed with MiniPar and a Syntactic Network representation was built from the dependency parse trees produced by the parser. Syntactic features were extracted from this SyntNet.

We considered two high-level Named Entity categories: Locations and Persons. For each of them five fine-grained sub-classes were taken into consideration. For locations: *mountain*, *lake*, *river*, *city*, and *country*; for persons: *statesman*, *writer*, *athlete*, *actor*, and *inventor*.

For each class under consideration we created a test set of Named Entities using WordNet 2.0 and Internet sites like Wikipedia. For the Class-Example approach we also provided training data using the same resources. WordNet was the primary data source for training and test data. The examples from it were extracted automatically. We

	P (%)	*R (%)*	*F (%)*
mountain	58	78	67
lake	75	50	60
river	69	55	61
city	56	76	65
country	86	93	89
locations macro	69	70	68
locations micro	78	78	78
statesman	42	72	53
writer	93	55	69
athlete	90	47	62
actor	90	73	80
inventor	12	33	18
persons macro	65	56	57
persons micro	57	57	57
total macro	67	63	62
total micro	65	65	65
category location	83	91	87
category person	95	89	92

Table 1: Performance of the Class-Example approach.

used Internet to get additional examples for some classes. To do this, we created automatic text extraction scripts for Web pages and manually filtered their output when it was necessary.

Totally, the test data comprised 280 Named Entities which were not ambiguous and appeared at least twice in the corpus.

For the Class-Example approach we provided a training set of 1194 names. The only requirement to the names in the training set was that they appear at least twice in the parsed corpus. They were allowed to be ambiguous and no manual post-processing or filtering was carried out on this data.

For both context feature approaches (i.e. Class-Word and Class-Example), we used the same type of syntactic features and the same classification function, namely the one described in Section 3.3. This was done in order to compare better the approaches.

Results from the comparative evaluation are shown in Table 2. For each approach we measured macro average precision, macro average recall, macro average F-measure and micro average F; for Class-Word and Class-Example micro F is equal to the overall accuracy, i.e. the percent of the instances classified correctly. The first row shows

	macro P (%)	*macro R (%)*	*macro F (%)*	*micro F(%)*
Class-Patterns	18	6	9	10
Class-Word	32	41	33	42
Class-Example	67	63	62	65
Class-Example (sec. ord.)	65	61	62	68

Table 2: Comparison of different approaches.

the results obtained with superficial patterns. The second row presents the results from the Class-Word approach. The third row shows the results of our Class-Example method. The fourth line presents the results for the same approach but using second-order features for the person category.

The Class-Pattern approach showed low performance, similar to the random classification, for which macro and micro F=10%. Patterns succeeded to classify correctly only instances of the classes "river" and "city". For the class "city" the patterns reached precision of 100% and recall 65%; for the class "river" precision was high (i.e. 75%), but recall was 15%.

The Class-Word approach showed significantly better performance (macro F=33%, micro F=42%) than the Class-Pattern approach.

The performance of the Class-Example (62% macro F and 65%-68% micro F) is much higher than the performance of Class-Word (29% increase in macro F and 23% in micro F). The last row of the table shows that second-order syntactic features augment further the performance of the Class-Example method in terms of micro average F (68% vs. 65%).

A more detailed evaluation of the Class-Example approach is shown in Table 1. In this table we show the performance of the approach without the second-order features. Results vary between different classes: The highest F is measured for the class "country" - 89% and the lowest is for the class "inventor" - 18%. However, the class "inventor" is an exception - for all the other classes the F measure is over 50%. Another difference may be observed between the Location and Person classes: Our approach performs significantly better for the locations (68% vs. 57% macro F and 78% vs. 57% micro F). Although different classes had different number of training examples, we observed that the performance for a class does not depend on the size of its training set. We think, that the variation in performance between categories is due to the different specificity of their textual contexts. As a consequence, some classes tend to co-occur with more specific syntactic features, while for other classes this is not true.

Additionally, we measured the performance of our approach considering only the macro-categories "Location" and "Person". For this purpose we did not run another experiment, we rather used the results from the fine-grained classification and grouped the already obtained classes. Results are shown in the last two rows of table 1: It turns out that the Class-Example method makes very well the difference between "location" and "person" - 90% of the test instances were classified correctly between these categories.

6 Conclusions and future work

In this paper we presented a new weakly supervised approach for Ontology Population, called Class-Example, and confronted it with two other methods. Experimental results show that the Class-Example approach has best performance. In particular, it reached 65% of accuracy, outperforming in our experimental framework the state-of-the-art Class-Word method by 42%. Moreover, for location names the method reached accuracy of 78%. Although the experiments are not comparable, we would like to state that some supervised approaches for fine-grained Named Entity classification, e.g. (Fleischman, 2001), have similar accuracy. On the other hand, the presented weakly supervised Class-Example approach requires as a training data only a list of terms for each class under consideration. Training examples can be automatically acquired from existing ontologies or other sources, since the approach imposes virtually no restrictions on them. This makes our weakly supervised methodology applicable on larger scale than supervised approaches, still having significantly better performance than the unsupervised ones.

In our experimental framework we used syntactic features extracted from dependency parse trees and we put forward a novel model for the representation of a syntactically parsed corpus. This model allows for performing a comprehensive extraction of syntactic features from a corpus including more complex second-order ones, which resulted in an improvement of performance. This and other empirical observations not described in this paper lead us to the conclusion that the performance of an Ontology Population system improves with the increase of the types of syntactic features under consideration.

In our future work we consider applying our Ontology Population methodology to more semantic categories and to experiment with other types of syntactic features, as well as other types of feature-weighting formulae and learning algorithms. We consider also the integration of the approach in a Question Answering or Information Extraction system, where it can be used to perform fine-grained type checking.

References

A. Almuhareb and M. Poesio. 2004. Attribute-based and value-based clustering: An evaluation. In *Proceedings of EMNLP 2004*, pages 158–165, Barcelona, Spain.

H. Avancini, A. Lavelli, B. Magnini, F. Sebastiani, and R. Zanoli. 2003. Expanding Domain-Specific Lexicons by Term Categorization. In *Proceedings of SAC 2003*, pages 793–79.

K. Bontcheva and H. Cunningham. 2003. The Semantic Web: A New Opportunity and Challenge for HLT. In *Proceedings of the Workshop HLT for the Semantic Web and Web Services at ISWC 2003.*

P. Buitelaar, P. Cimiano, and B. Magnini, editors. 2005. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, Amsterdam, The Netherlands.

P. Cimiano and J. Völker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of RANLP'05*, pages 166–172, Borovets, Bulgaria.

P. Cimiano, A. Pivk, L.S. Thieme, and S. Staab. 2005. Learning Taxonomic Relations from Heterogeneous Sources of Evidence. In *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press.

M. Fleischman and E. Hovy. 2002. Fine Grained Classification of Named Entities. In *Proceedings of COLING 2002*, Taipei, Taiwan, August.

M. Fleischman. 2001. Automated Subcategorization of Named Entities. In *39th Annual Meeting of the ACL, Student Research Workshop*, Toulouse, France, July.

M. Hearst. 1998. Automated Discovery of WordNet Relations. In *WordNet: An Electronic Lexical Database*. MIT Press.

D. Lin. 1998a. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of COLING-ACL98*, Montreal, Canada, August.

D. Lin. 1998b. Dependency-based Evaluation of MiniPar. In *Proceedings of Workshop on the Evaluation of Parsing Systems*, Granada, Spain.

S. Schlobach, M. Olsthoorn, and M. de Rijke. 2004. Type Checking in Open-Domain Question Answering. In *Proceedings of ECAI 2004.*

I. Szpektor, H. Tanev, I. Dagan, and B. Coppola. 2004. Scaling Web-based Acquisition of Entailment Relations. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

P. Velardi, R.Navigli, A. Cuchiarelli, and F.Neri. 2005. Evaluation of Ontolearn, a Methodology for Automatic Population of Domain Ontologies. In P. Buitelaar, P. Cimiano, and B. Magnini, editors, *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press.