From Web Directories to Ontologies: Natural Language Processing Challenges[⋆]

Ilya Zaihrayeu[1], Lei Sun[2], Fausto Giunchiglia[1], Wei Pan[2], Qi Ju[3,1], Mingmin Chi[2], Xuanjing Huang[2]

[1] University of Trento, Italy
{ilya, fausto}@dit.unitn.it
[2] Fudan University, China
{leisun, panwei, mmchi, xjhuang}@fudan.edu.cn
[3] Jilin University, China
qi@jlu.edu.cn

Abstract. Hierarchical classifications are used pervasively by humans as a means to organize their data and knowledge about the world. One of their main advantages is that natural language labels, used to describe their contents, are easily understood by human users. However, at the same time, this is also one of their main disadvantages as these same labels are ambiguous and very hard to be reasoned about by software agents. This fact creates an insuperable hindrance for classifications to being embedded in the Semantic Web infrastructure. This paper presents an approach to converting classifications into lightweight ontologies, and it makes the following contributions: (i) it identifies the main NLP problems related to the conversion process and shows how they are different from the classical problems of NLP; (ii) it proposes heuristic solutions to these problems, which are especially effective in this domain; and (iii) it evaluates the proposed solutions by testing them on DMoz data.

1 Introduction

The success of the Web was particularly conditioned by the ease with which its users could publish and interlink their data. However, as the Web has grown larger, it has become essential to categorize the huge amounts of documents on the web. Hierarchical classifications, whose nodes are assigned natural language labels, perfectly serve this purpose. In fact, there are plenty of classifications on the web: web directories like DMoz[4], business catalogues like Amazon[5], topic categories like Wikipedia[6], site maps in web portals and in personal pages are examples that demonstrate the pervasive presence of classifications on the web.

[⋆] This work has been partly supported by the OpenKnowledge project (FP6-027253, see http://www.openk.org) and by Natural Science Foundation of China under grant No. 60673038.

[4] See http://www.dmoz.org.

[5] See http://www.amazon.com.

[6] See http://www.wikipedia.org.

The underlying idea of the Semantic Web is that web content should be expressed not only in natural language, but also in a language that can be unambiguously understood, interpreted and used by software agents, thus permitting them to find, share and integrate information more easily [3]. The cental notion to this idea is *ontology*, which defines a taxonomy of classes of objects and relations among them [3]. Differently from classifications, ontologies should be written in a formal language such as OWL [16], which is unambiguous and suitable for being reasoned about by software agents.

Ontologies are very hard to be designed by an ordinary user of the Web, and designing an OWL-DL [16] ontology is a difficult and error-prone task even for experienced users [22]. This fact further complicates a classic chicken-and-egg problem which prevents the Semantic Web from scaling in the large: users will not mark up their data unless they perceive an added value from doing so, and tools to demonstrate this value will not be developed unless a "critical mass" of annotated data is achieved [12]. As Hendler further remarks in [12], "*Lowering the cost of markup isn't enough – for many users it needs to be free. That is, semantic markup should be a by-product of normal computer use*".

On the other hand, classifications are very easy to be created and maintained by an ordinary user. They represent a very natural way for (natural language) markup of the data classified in them. Moreover, classifications are used pervasively on the web thus creating the necessary "critical mass" of annotated data. These facts seem to resolve the chicken-and-egg problem. However, because they are described in natural language, classifications cannot be easily embedded in the infrastructure of the Semantic Web. To address this problem, [9] discusses how classifications can be scaled up to the Semantic Web by converting them into lightweight ontologies, and [11] demonstrates the practical applicability of the approach in its application to automatic ontology-based document classification.

The current paper extends the work presented in [9, 11] by analyzing in detail the principle step of conversion from natural language to formal language. The main natural language processing (NLP) problems related to the conversion process are: named entity (NE) locating, part-of-speech (POS) tagging, word sense disambiguation (WSD), and parsing. We show how these problems, when applied to the classification domain, are different from their classical application on full-fledged sentences. We propose heuristic solutions to the NE locating, POS tagging, and WSD problems, and evaluate their performance by testing them on DMoz data. As we show in the paper, NE locating is a much easier problem in the DMoz data set, where we reach 93.45% of precision; in a POS tagging task we reach 96.00% of precision which is 11.52% higher than in the application of the POS tagger trained on full-fledged sentences; and, in the WSD task we reach 66.51% of accuracy which is an acceptable performance result according to the state-of-the-art in this field of NLP.

The paper is organized as follows. In Section 2 we discuss how we convert classifications into lightweight ontologies and show how the above mentioned NLP problems are relevant to this conversion process. Sections 3, 4, and 5 discuss particular problems of, proposed solutions and evaluation results for NE locating, POS tagging, and WSD respectively. In Section 6 we discuss the related work. Section 7 summarizes the results and concludes the paper.

2 From classifications to lightweight ontologies

Classification labels are expressed in natural language, which is ambiguous and very hard to be reasoned about. In order to address this problem, we encode classification labels into formulas in propositional Description Logic language L^C, following the approach described in [9]. Note that even if L^C is propositional in nature, it has a set-theoretic semantics. Namely, the interpretation of a (lexically expressed) concept is the set of documents, which are about this concept [9]. For instance, the interpretation of concept `Capital` (defined as "a seat of government") is the set of documents about capitals, and *not* the set of capitals which exist in the world. Below we briefly describe how we encode classification labels into formulas in L^C. Interested readers are referred to [9] for a complete account. Here, we discuss the conversion process in a limited extent while focusing on the related NLP problems.

WordNet [17] senses of adjectives and common nouns become atomic concepts. The extension of a common noun concept is the set of documents about objects of the class, denoted by the noun; and, the extension of an adjective concept is the set of documents about objects, which possess the qualities, denoted by the adjective. Proper names (also recognized as named entities) become atomic concepts as well, whose extension is the set of documents about the individual referenced by the proper name. Notationally, we construct adjective and common noun atomic concepts using the following syntax: *lemma-pos-sn*, where *lemma* is the lemma of the word, *pos* is its part of speech, and *sn* is the sense number in WordNet [17]. We use $_{NNP}$ to mark proper name atomic concepts.

Atomic concepts are then connected to form complex concepts as follows: syntactic relations between words are translated to logical connectives of L^C. For example, a set of adjectives followed by a noun group is translated into the logical conjunction ($\sqcap$) of the concepts corresponding to the adjectives and to the nouns; prepositions like "of" and "in" are translated into the conjunction; coordinating conjunctions "and" and "or" are translated into the logical disjunction ($\sqcup$). The final formula for a label is built following these rules and taking into account how words are coordinated in the label. The final formulas are then assigned to classification nodes, thus converting the classification into a lightweight ontology. These ontologies can be used for automating various tasks on classifications, such as semantic search [9], semantic matching [10], and document classification [11].

Let us consider a relatively complex label: "*Bank and personal details of George Bush*". Its correct translation to L^C will produce the following concept:

$$(\texttt{bank-noun-1} \sqcup \texttt{personal-adj-1}) \sqcap \texttt{detail-noun-1} \sqcap \texttt{george\_bush}_{NNP}$$

The extension of the concept above is the intersection of three sets of documents: (i) documents about the President George W. Bush, (ii) documents containing isolated facts about something (i.e., details), and (iii) the union of documents about bank institutions and documents concerning a particular person or his/her private life. As it can be seen, the extension includes documents one would classify under a node with the above given natural language label.

Despite its seeming simplicity, the translation process is subject to various mistakes originating from incorrect NLP. For instance, due to a mistake in POS tagging, the word *personal* might be recognized as a noun, which has only one sense in WordNet defined as "*a short newspaper article about a particular person or group*"; due to a mistake in WSD, the word *bank* might be identified as "*sloping land (especially the slope beside a body of water)*"; due to a mistake in NE locating, the proper name *George Bush* might not be recognized and might then be considered as two distinct nouns, where the noun *bush* means "*a low woody perennial plant usually having several major branches*"; finally, due to a mistake in (syntax) parsing, the input label might be translated into:

$$\texttt{bank-noun-1} \sqcup \texttt{personal-adj-1} \sqcap \texttt{detail-noun-1} \sqcap \texttt{george\_bush}_{NP}$$

a concept, whose extension is the union of documents about bank institutions and documents discussing personal details of the President George W. Bush.

The NLP framework, which enables the conversion of classification labels into formulas in L^C, is depicted in Fig. 1. It follows the standard NLP pipeline: tokenization, NE locating, POS tagging, WSD, and parsing. In our framework we assume that a label is either an NE or a non-NE. Therefore, NE labels skip the last three steps since they already represent atomic concepts.

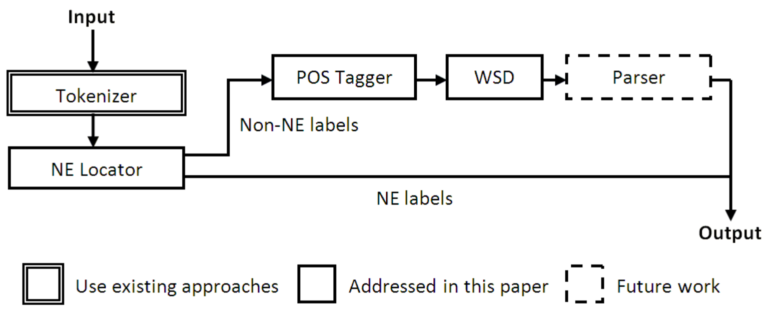


Fig. 1. Framework of NLP on a Web directory.

Tools developed for general NLP tasks cannot be directly used in our framework. One main reason is that the corpus we use, namely, labels in Web directories, is significantly different from those on which most NLP tools are developed. These differences are briefly described as follows:

– Web directory labels are short phrases, while general NLP tools are developed on full-fledged sentences;

- Most of the words in a Web directory are nouns, adjectives, articles, conjunctions and prepositions. The verbs and pronouns are very rare in a Web directory while being common in full-fledged sentences;
- NEs occur densely in a Web directory. This is not surprising, as a Web directory is a knowledge base, which unavoidably has many proper nouns that describe entities in the world;
- The capital rule is different in a Web directory. In full-fledged sentences, the first words of sentences and the words in proper names are initialized with capital letters. In a Web directory, however, most often every word begins with a capital letter except for prepositions and conjunctions;
- The proper sense of a word may depend on the meaning of a word appearing in a label located higher in the classification tree. For instance, noun "Java" means an island if it appears under a node with label "Geography".

In this paper, we focus on NE locating, POS tagging and WSD on a Web directory. We perform tokenization following the standard approach from Penn Treebank [18], and we leave parsing to the future work as how to do it strongly depends on the results presented in this paper.

3 Named entity recognition

The data set we used for our analysis and evaluation (of the NE locator and the POS taggers) is built on DMoz. According to a dump created in 2004, the DMoz directory has 698,057 labels (nodes). There are many non-English labels, which can be excluded from consideration by discarding the DMoz subtrees rooted at the following nodes: `Top/World`, `Top/Kids_and_Teens/International`, and `Top/Adult/World`. As the result, we have 474,389 English labels. For these English labels, the average length (i.e., the number of tokens) is 1.91 and the average depth (i.e., the number of hops from the root) is 7.01.

Out of 474,389 labels we randomly selected 12,365 labels (2.61%) for analysis and manual annotation. Each label in this data set has been annotated with POS and NE information. As the result, we have totally 8177 non-NE labels (66.13%) and 4188 NE labels (33.87%). We observed that nearly all NEs take *entire* labels. We manually examined the data set and found only 7 exceptional labels (0.06%). Therefore, the assumption made in our NLP framework is valid. In Table 1 we report statistics of POS occurrences in the non-NE labels in our data set.

3.1 The approach

By analyzing the data set, we noticed the following characteristics of NEs:

- Rare labels tend to be NEs. A general label (such as `Arts and Entertainment`) can occur thousands of times in a Web directory, while NE labels occur much more rarely. Most of NE labels, such as "`Schindler's List`" (a movie name) occur only once;

POS	`NN`	`NNS`	`CC`	`JJ`	`NNP`	`IN`	`,`	`TO`	`CD`
Occurrence	7714	3619	2893	1020	239	235	72	18	11
Percentage	48.68	22.84	18.26	6.44	1.51	1.48	0.45	0.11	0.07
POS	`:`	`VBN`	`DT`	`RB`	`POS`	`VB`	`NPS`	`JJR`	
Occurrence	9	6	4	2	2	1	1	1	
Percentage	0.06	0.04	0.03	0.01	0.01	< 0.01	< 0.01	< 0.01	

Table 1. Statistics of POS occurrences in the data set.

– Labels in which most of the tokens are rare words tend to be NEs, e.g., `Agios Dometios` is a geography name and each of its tokens occurs only once in the whole directory;
– There are so-called letter bars in Web directories, such as single letter “A”, “B”, ..., “Z” and also double letters “Aa”, “Ab”, ..., “Zz”. These labels are created only for the convenience of navigation. Besides, they are good indicators of NEs, as nearly all children of these labels are NEs;
– In an NE label, initial articles, such as “the”, “a” and “an”, are usually put at the end after a comma. For example, “`The Magic School Bus`” is written as “`Magic School Bus, The`”. This is another good indicator of NEs;
– The NE and non-NE labels distribute differently on their lengths. This difference is illustrated in Fig. 2(a), which is the statistical result of label length in our data set;
– The NE and non-NE labels distribute differently on their depths. This difference is illustrated in Fig. 2(b), which is the statistical result of label depth in our data set.

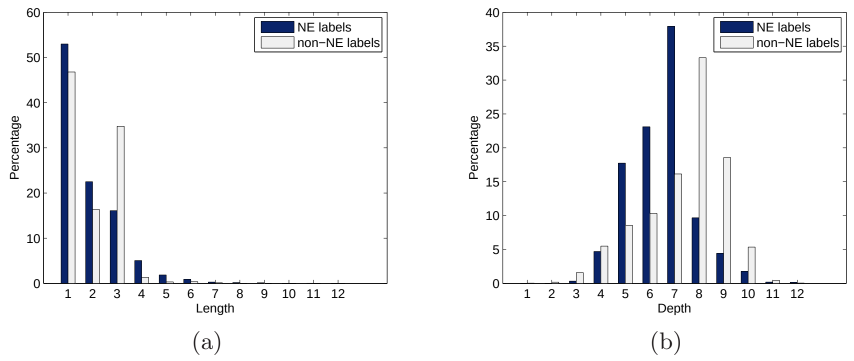


Fig. 2. (a) Label length distribution; (b) Label depth distribution.

Taking these characteristics into account, we implemented the NE locator using Conditional Maximum Entropy Model (CMEM) [2] with Gaussian smoothing [6]. The features for CMEM have been chosen according to the characteristics described above. Particularly, we consider the following feature classes (i.e., sets of features) in the implementation:

- *WordsInLabel*: The first two and the last two tokens in the label;
- *WordsInPath*: The first and the last tokens in the label's parent, grandparent, the farthest ancestor (excluding the root "`Top`") and the second farthest ancestor;
- *LengthOfLabel*: The number of tokens in the label;
- *DepthOfLabel*: Depth of the label (distance from the root node);
- *FrequencyOfLabel*: Count how many times the label occurs in the whole directory;
- *AveFrequencyOfTokens*: Count how many times each token in the label occurs in the whole directory, and calculate the average.

To make it clearer, in Table 2 we show features which are extracted from a label located on the following path:

```
Top/Computers/Internet/Chat/Instant Messaging/AOL Instant
                         Messenger
```

Feature class: *WordsInLabel*	
First token of the current label	`AOL`
Second token of the current label	`Instant`
Last token of the current label	`Messenger`
Second last token of the current label	`Instant`
Feature class: *WordsInPath*	
First token of the parent	`Instant`
Last token of the parent	`Messaging`
First token of the grandparent	`Chat`
Last token of the grandparent	`Chat`
First token of the farthest ancestor	`Computers`
Last token of the farthest ancestor	`Computers`
First token of the second farthest ancestor	`Internet`
Last token of the second farthest ancestor	`Internet`
Feature class: *LengthOfLabel*	
Length of the current label	3
Feature class: *DepthOfLabel*	
Depth of the current label	5
Feature class: *FrequencyOfLabel*	
Frequency of the current label	1
Feature class: *AveFrequencyOfTokens*	
Average frequency of tokens of the current label	$(13 + 8 + 6)/3 = 9$

Table 2. An Example of Features Extracting for NE Recognizer

3.2 Evaluation

This experiment is performed in two steps. First, we train the NE locator by using each feature class to compare their contributions. Then, we train the NE locator again with some combinations of feature classes to see the best performance we can reach. To make our experimental results more reliable, we perform 6-fold cross validation. We use the following 3 measures to evaluate the performance of the NE locator:

- **Precision of NE locating (PNE)**. We count how many labels picked out by the NE locator are real NE labels (those annotated as NEs in the data set), and calculate the percentage;
- **Recall of NE locating (RNE)**. We count how many real NE labels are picked out by the NE locator, and calculate the percentage;
- **F-score of NE locating (FNE)**. An overall measure of performance of the NE locator, which combines PNE and RNE as:

$$\mathrm{FNE} = \frac{2 \cdot \mathrm{PNE} \cdot \mathrm{RNE}}{\mathrm{PNE} + \mathrm{RNE}}$$

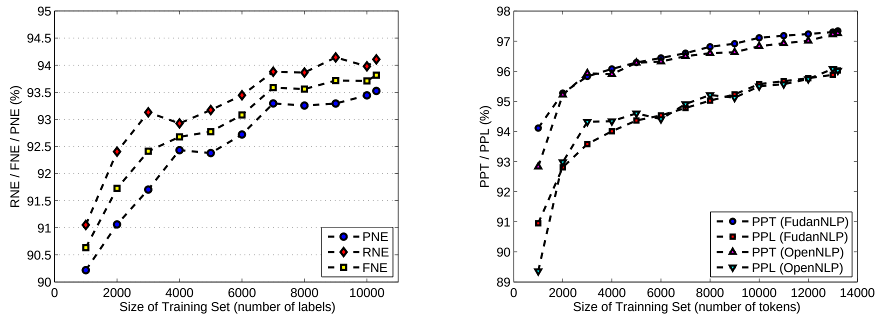
We report the performance results of the NE locator in Table 3. As it can be observed, feature classes *WordsInLabel* and *WordsInPath* provide the most important contributions to the precision. By combining these two feature classes we can get the performance which is close to that provided by combining all the feature classes (compare the figures reported in row "1+2" with ones in row "1+2+3+4+5+6").

Feature Class	PNE	RNE	FNE
1. *WordsInLabel*	81.49	94.33	87.45
2. *WordsInPath*	89.48	79.36	84.12
3. *FrequencyOfLabel*	75.04	91.30	82.37
4. *AveFrequencyOfTokens*	76.05	82.95	79.35
5. *DepthOfLabel*	53.13	78.76	63.45
6. *LengthOfLabel*	64.64	8.05	14.32
1+2	92.08	94.20	93.13
1+2+3+4+5+6	93.45	94.04	93.75

Table 3. Performance results of the NE locator.

One state-of-the-art system [8] of NE locating in the Web environment on full-fledged sentences has the performance of 59% in precision, 66% in recall and 38% in F-score. Similar to our task, NE locating in the Web environment share the difficulty of large amount of undefined entity classes. The reason our approach outperforms theirs is that NE locating on Web directories is a relatively easy task, as we only need to tell whether a label is an NE or not.

To check whether our data set is properly sized, we performed incremental training, namely, keeping the testing set unchanged, we checked how performance



(a) (b)

Fig. 3. Incremental training of: (a) NE locator; (b) POS Taggers.

varied with the growing size of the training set. In Fig. 3(a) we show the achieved results. As it can be observed, PNE, RNE, and FNE increase significantly when the size of the training set grows from 1000 to 7000 samples. When the number of samples becomes greater than 7000, the performance measures change slightly. Empirically, we conclude that our NE locating model is effective and stable enough to be used on web directories such as DMoz.

4 Part of speech tagging

4.1 The approach

Nearly all state-of-the-art POS taggers are based on supervised learning approaches. In these approaches, first, a properly sized manually annotated corpus is created. Then, a model is trained on this corpus to allow for further POS tagging of unseen data. Popular models for POS tagging include Hidden Markov Model (HMM) [21], Conditional Maximum Entropy Model (CMEM) [2], and Conditional Random Field (CRF) [15]. Below we briefly describe CMEM and CRF, as they are used by the POS taggers we employ in our experiments.

To tag a token, CMEM considers the context of the token by introducing the notion of feature. In the task of POS tagging, the context of a token is usually the token itself and its surroundings. A feature is a function which maps the context of the token to a 0-1 value. Namely, it answers a yes/no question about the context. CMEM learns how POS is conditioned by contexts as a set of probability distributions from a manually annotated corpus. The learning process applies a max-entropy style parameter optimization. CMEM tags the tokens sequentially (starting from the left-most token in the sentence) by assigning the POS with the highest conditional probability to each token given the token's context.

Differently from CMEM, in CRF, the POS of a token is conditioned by contexts of *all* the tokens in the given sentence. This allows for a global coordination among local taggings. This property makes CRF a more advanced model for the POS tagging task.

In our experiments, we employed two POS taggers: the CRF-based FudanNLP POS tagger [20] and the CMEM-based OpenNLP POS tagger [19]. We retrained these tools on our data set and checked if we gain an improvement in accuracy w.r.t. the case when the tools are trained on full-fledged sentences. To avoid a negative influence of NE labels on the training of a POS tagger, both POS taggers were trained and tested only on the non-NE labels in the data set.

4.2 Evaluation

To make our experimental results more reliable, we perform 6-fold cross validation. The following 2 measures are used to evaluate the performance of the POS taggers:

- **Precision of POS tagger by Tokens (PPT)**. The granularity of this precision measure is a token, namely, we count tokens which are tagged with the correct tag, and calculate the percentage;
- **Precision of POS tagger by Labels (PPL)**. The granularity of this precision measure is a label, namely, we count labels whose tokens are all correctly tagged, and calculate the percentage.

The evaluation results are shown in Table 4, where the following notations are used: PPT_0 and PPL_0 refer to the PPT and PPL before retraining, while PPT_1 and PPL_1 refer to the PPT and PPL after retraining. Note that there is no significant difference in the performance between the CMEM approach (OpenNLP) and the CRF approach (FudanNLP). It has been proven that CRF outperforms CMEM when tagging full-fledged sentences, since CRF considers global coordination in a sentence while CMEM only considers local context. However, in DMoz, labels (symmetric to sentences) are too short (1.91 tokens on the average). In most cases, CMEM features of a single token are able to consider information of the whole label. In other words, CMEM is able to do something like global coordination as CRF on these short labels. This property of our data set makes CRF similar to CMEM in performance.

The state-of-the-art performance of a POS tagger on full-fledged sentences is 97.24% in token precision (PPT) according to [24], which is very close to ours. However, precision by sentences should be lower than our PPL, as our labels are much shorter.

	PPT_0	PPT_1	Gain	PPL_0	PPL_1	Gain
OpenNLP	91.27	97.23	+6.16	84.68	96.00	+11.52
FudanNLP	96.12	97.33	+1.21	92.72	96.02	+3.30

Table 4. Performance results of the OpenNLP and FudanNLP POS taggers before and after retraining.

We performed incremental training of the POS taggers, too. The result is shown in Fig. 3(b), which demonstrates a trend similar to that in Fig. 3(a). Empirically, we conclude that our POS tagging model is effective and stable enough to be used on web directories such as DMoz.

5 Word sense disambiguation

5.1 The approach

The proposed WSD algorithm traverses the nodes of the classification tree in the BFS or DFS order. Then, at each node, it first finds *concept tokens*, i.e., tokens which are present in WordNet as adjectives and/or as nouns. Next, it identifies *ambiguous concept tokens*, i.e., concept tokens which have more than one sense. Ambiguous concept tokens of each node are processed by the algorithm following the steps reported below. If a token is not disambiguated at step n, then it is processed at step $n + 1$. The ultimate goal of the algorithm is to select only one sense for each ambiguous concept token. Below we say that a token sense is *active* if it has not been discarded.

1. Identify the POS of the token and, if the token has senses of this POS, then preserve these senses and discard senses belonging to the other POS, if any;
2. Preserve noun token senses if they are hypernyms or hyponyms of active noun senses of other concept tokens in the label, and discard the other senses. Hypernymy and hyponymy relations amongs noun token senses are checked using the WordNet hypernymy hierarchy [17];
3. Preserve noun token senses if they are located within a certain distance in the WordNet hypernymy hierarchy from active noun senses of other concept tokens in the label. If there are several matching senses with different distances, then preserve those with the shortest distance and discard the others;
4. Preserve noun token senses if they are hyponyms of active noun senses of concept tokens appearing in the label of an ancestor node, and discard the other senses. Note that we do not consider hypernyms since, as reported in [11], higher level nodes usually represent more general categories than lower level nodes;
5. Preserve noun token senses if they are located within a certain distance in the WordNet hypernymy hierarchy from active noun senses of concept tokens appearing in the labels of ancestor nodes, and discard the other senses. If there are several matching senses with different distances, then preserve those with the shortest distance and discard the others;
6. Preserve the first active noun sense (in WordNet) and discard the other active senses. If there is no active noun sense, then preserve the first active adjective sense and discard the other active senses. Noun senses prevail over adjective senses since, according to the results reported in Table 1, nouns are much more frequent in a web directory than adjectives. Note that senses in WordNet are sorted in the descendant order of the number of times they

were tagged in the semantic concordance texts [17]. Therefore, picking up a sense that appears higher in the list of senses increases the probability of that the sense will be the correct meaning of the token. After this step, the token is disambiguated.

5.2 Evaluation

To evaluate the performance of our WSD algorithm, we have selected a DMoz subtree rooted at `Top/Business/Consumer_Goods_and_Services`. The subtree has 781 nodes, its maximal depth is 6, and the average branching factor is 4.22. Its nodes have 1368 tokens in total. There are 1107 concept tokens, out of which 845 are ambiguous. The average polysemy of an ambiguous concept token is 4.05. Note that this data set is different from that used in NE locating and POS tagging because the WSD algorithm requires a POS tagged subtree and not a set of randomly selected labels. The two data sets do not have nodes in common, which ensures unbiased performance of the POS tagger at step 1 of the algorithm.

In Table 5 we report the results of the algorithm measured in 8 experiments. For each step we provide the number of disambiguated tokens and the accuracy of the step. For steps 3 and 5 we provide the similarity distance threshold as the number of edges in the WordNet hypernymy hierarchy. We write "na" (not applicable) as a parameter and an output value of a step when the step was skipped in the experiment. In the right-most column we report the overall accuracy. At step 1 of the algorithm, we used the FudanNLP POS tagger, which was beforehand trained on the whole data set described in Section 3. At steps 2-5 we used WordNet version 2.1.

#	Step 1		Step 2		Step 3			Step 4		Step 5			Step 6		Accur.
	tok.	acc.	tok.	acc.	thr.	tok.	acc.	tok.	acc.	thr.	tok.	acc.	tok.	acc.	
1	na	na	na	na	na	na	na	na	na	na	na	na	845	63.90	63.90
2	84	98.81	na	na	na	na	na	na	na	na	na	na	761	60.84	64.62
3	84	98.81	11	100	na	na	na	na	na	na	na	na	750	61.60	65.80
4	84	98.81	11	100	10	250	50.80	na	na	na	na	na	500	61.20	62.37
5	84	98.81	11	100	2	24	87.50	na	na	na	na	na	726	60.88	65.92
6	84	98.81	11	100	2	24	87.50	8	87.50	na	na	na	718	61.28	66.51
7	84	98.81	11	100	2	24	87.50	8	87.50	10	379	33.24	339	42.77	46.51
8	84	98.81	11	100	2	24	87.50	8	87.50	2	43	41.86	675	60.00	64.49

Table 5. Performance results of the WSD algorithm.

The baseline solution, i.e., when only step 6 is executed, gives 63.90% accuracy. Step 1 performed reasonably well, correctly disambiguating 98.81% of about 10% of tokens. Step 2 disambiguated a small number of tokens (11) but all of them were disambiguated correctly. Step 3 performed reasonably well on small thresholds producing the best result when the threshold value was 2 (compare step 3 in experiments 4 and 5). A similar trend can be observed for step 5. However, even with the threshold value of 2, its accuracy is lower than the baseline accuracy, thus making a negative effect on the overall accuracy. Note that when the threshold value is 2, steps 3 and 5 preserve senses which are siblings in the WordNet hypernymy hierarchy. The best performance of the algorithm was recorded in experiment 6 with the accuracy value of 66.51%, which is 2.61% higher than the baseline.

The performance of a state-of-the-art WordNet-based WSD algorithm on full-fledged sentences varies according to the underlying approach. For instance, the best accuracy of the WSD algorithm presented in [1] is 47.3% for polysemous nouns. Similar to our case, in [1] the best accuracy is only slightly higher than the baseline. A more recent work, [25], uses a web search engine (together with WordNet) for WSD and reaches 76.55% accuracy for polysemous nouns in the best case. While the average polysemy of nouns is close to ours (4.08), the size of the context window varied from 3 to 7 words that are known to WordNet, what is not possible to have in our case. Empirically, we conclude that the result of our WSD algorithm is comparable to the state-of-the-art in this field of NLP.

6 Related work

It is now quite clear that there is an insuperable gap between the logic-based Semantic Web and its real-world users because it is nearly impossible for an ordinary user to learn (how to use) an unfamiliar formal language [5, 14, 12, 23]. To address this problem, ontology authoring (e.g., see [4, 23]), interfacing (e.g., see [7]), and querying (e.g., see [5]) tools that use natural language as an interface with the human user have been proposed. The approach to converting classifications into lightweight ontologies, described in this paper, shares the spirit of [12, 14, 5, 4, 7, 23] and makes a step exactly in the direction of helping users to benefit from the Semantic Web without requiring them to go through the burdensome learning curve.

We differ from the above cited approaches in several respects. For instance, the approaches reported in [4, 5, 7] require an ontology at the backend against which the user could formulate her queries in natural language. To reduce both ambiguity and complexity of natural language, [4, 5, 23] use a controlled language for user queries, therefore requiring the user to know about the used subset of English, its semantics and grammar rules. In order to provide meaningful input, in [23] the user still needs to understand the general principles of first order logic. None of these requirements are made in our approach, in which the user does not need to have preloaded ontologies at the backend. Instead, the user creates an ontology seamlessly as *a by-product of normal computer use* – by creating a classification. The complexity of language is lower and no controlled language is required – the user naturally uses noun phrases to describe classification nodes. Noteworthy, even if our approach is more lightweight, it still allows for automating various tasks on classifications, such as semantic search [9], semantic matching [10], and document classification [11].

The approach described in [13] allows it to convert a hierarchical classification into an OWL ontology by deriving OWL classes from classification labels and by arranging these classes into a hierarchy (based on the `rdfs:subclassOf` relation) following the classification structure. The approach is based on some application-dependent assumptions such as that one label represents one atomic concept, and that relations between labels can be defined as is-a relations in some context (e.g., concept "ice" is more specific than concept "non-alcoholic beverages" when considered in the context of procurement [13]). These assumptions do not hold in a general case and are not made in our approach. More importantly, in the current paper we provide a complete account of the NLP problems which need to be dealt with when converting classifications into ontologies. This problem is not addressed in [13] and can be seen as a preliminary step to their work.

7 Conclusions

The paper presents an approach to converting classifications into lightweight ontologies and discusses in detail the NLP problems related to this conversion process on the example of the DMoz web directory. The NLP analysis reported in this paper, to the best of our knowledge, is the first investigation of how NLP technology can be applied on classification labels and, more generally, on short natural language (noun) phrases. Noteworthy, even if the application domain we consider is different from the one on which NLP technology is usually applied, the results reported in this paper are comparable with (and, sometimes, exceeding) those reached by the state-of-the-art NLP tools.

References

1. E. Agirre and G. Rigau. A proposal for word sense disambiguation using conceptual distance. In *the First International Conference on Recent Advances in NLP*, Tzigov Chark, Bulgaria, September 1995.
2. A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. Maximum entropy approach to natural language processing. *Computational Linguistic*, 22(1):39–71, 1996.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, (284(5)):34–43, May 2001.
4. A. Bernstein and E. Kaufmann. GINO - a guided input natural language ontology editor. In *International Semantic Web Conference*, pages 144–157, 2006.
5. A. Bernstein, E. Kaufmann, A. Göhring, and C. Kiefer. Querying ontologies: A controlled English interface for end-users. In *International Semantic Web Conference*, pages 112–126, 2005.
6. S. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS -99-108, Carnegie Mellon University, 1999.
7. W. Chong, X. Miao, Z. Qi, and Y. Yong. PANTO: A Portable Natural Language Interface to Ontologies. In *Proceedings of the European Semantic Web Conference*, volume 4519 of *Lecture Notes in Computer Science*. Springer-Verlag, July 2007.

8. D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *Proc. of IJCAI, 2007*, 2007.
9. F. Giunchiglia, M.Marchese, and I. Zaihrayeu. Encoding classifications into lightweight ontologies. In *JoDS VIII: Special Issue on Extended Papers from 2005 Conferences*, Winter 2006. A shorter version of the paper appeared in ESWC-2006.
10. F. Giunchiglia, M. Yatskevich, and P. Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics*, IX, 2007.
11. F. Giunchiglia, I. Zaihrayeu, and U. Kharkevich. Formalizing the get-specific document classification algorithm. In *11th European Conference on Research and Advanced Technology for Digital Libraries*, Budapest, Hungary, September 2007.
12. J. Hendler. Agents and the semantic web. *IEEE Intelligent Systems*, (2), 2001.
13. M. Hepp and J. de Bruijn. GenTax: A generic methodology for deriving owl and RDF-S ontologies from hierarchical classifications, thesauri, and inconsistent taxonomies. In *4th European Semantic Web Conference*, LNCS 4519, pages 129–144, Innsbruck, Austria, June 3-7 2007. Springer.
14. B. Katz and J. Lin. Annotating the semantic web using natural language. In *NLPXML '02: Proceedings of the 2nd workshop on NLP and XML*, pages 1–8, Morristown, NJ, USA, 2002. Association for Computational Linguistics.
15. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of 18th International Conf. on Machine Learning*, 2001.
16. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C), February 10 2004.
17. G. Miller. *WordNet: An electronic Lexical Database.* MIT Press, 1998.
18. Penn Treebank Project. See http://www.cis.upenn.edu/ treebank/.
19. The OpenNLP project. See http://opennlp.sourceforge.net/.
20. X. Qian. A CRF-based pos tagger. Technical Report FDUCSE 07302, Fudan University, 2007.
21. L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proc. of the IEEE*, volume 77(2), pages 257–285, 1989.
22. A. L. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In *EKAW*, pages 63–81, 2004.
23. R. Schwitter and M. Tilbrook. Lets talk in Description Logic via controlled natural language. In *Logic and Engineering of Natural Language Semantics (LENLS2006)*, Tokyo, Japan, 2006.
24. K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1, pages 173–180, 2003.
25. C. Yang and J. C. Hung. Word sense determination using wordnet and sense co-occurrence. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 1*, pages 779–784, Washington, DC, USA, 2006. IEEE Computer Society.