Lexical patterns or dependency patterns: which is better for hypernym extraction?

Tjong Kim Sang, E.; Hofmann, K.

Publication date
2009

Document Version
Final published version

Published in
Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)

Lexical Patterns or Dependency Patterns: Which Is Better for Hypernym Extraction?

Erik Tjong Kim Sang
Alfa-informatica
University of Groningen
e.f.tjong.kim.sang@rug.nl

Katja Hofmann
ISLA, Informatics Institute
University of Amsterdam
khofmann@science.uva.nl

Abstract

We compare two different types of extraction patterns for automatically deriving semantic information from text: lexical patterns, built from words and word class information, and dependency patterns with syntactic information obtained from a full parser. We are particularly interested in whether the richer linguistic information provided by a parser allows for a better performance of subsequent information extraction work. We evaluate automatic extraction of hypernym information from text and conclude that the application of dependency patterns does not lead to substantially higher precision and recall scores than using lexical patterns.

1 Introduction

For almost a decade, automatic sentence parsing systems with a reasonable performance (90+% constituent precision/recall) have been available for English (Charniak, 1999). In recent years there has been an increase in linguistic applications which use parsing as a preprocessing step, e.g. Snow et al. (2006) and Surdeanu et al. (2008). One of the boosts for these new applications was the increasing power of desktop computers, which allows for an easier access to the computing-intensive parsing results. Another is the increased popularity of dependency parsing of which the results can easily be incorporated into followup systems.

Although there is a consensus about the fact that the richness of the dependency structures should, in principle, enable better performance than lexical information or shallow parsing results, it is not clear if these better results can also be obtained in practice. A performance of 90% precision and recall at constituent level still leaves an average of one error in a medium-length sentence of ten words. These errors could degrade the performance of any approach which relies heavily on parser output.

The question of whether to include a full parser as a preprocessor for natural language processing task, has led to a heated discussion between the two authors of the paper. One of us argues that full parsers are slow and make too many errors, and relies on shallow techniques like part-of-speech tagging for preprocessing. The other points at the decreasing costs of computing and improvements in the reliability of parsers, and recommends dependency parsers as preprocessing tools.

While no automatic text preprocessing method is free of errors, it is indeed true that approaches other than full parsing, like for example shallow parsing, offer useful information at a considerably cheaper processing cost. The choice between using a heavy full parser or a light shallow language analyzer is one that developers of language processing systems frequently have to make. The expected performance boost of parsed data could be an important motivation for choosing for full syntactic analysis. However, we do not know how big the difference between the two methods will be. In order to find this out, we designed an experiment in which we compared the effects of preprocessing with and without using information generated by a full parser.

In this paper, we compare two text preprocessing approaches for a single language processing task. The first of the two methods is shallow lin-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 174–182,
Boulder, Colorado, June 2009.

guistic processing, a robust and fast text analysis method which only uses information from words, like lemma information and part-of-speech classes. The second method is dependency parsing which includes information about the syntactic relations between words. The natural language processing task which we will use for assessing the usability of the two processing methods is automatic extraction of hypernym information from text. The language of the text documents is Dutch. We expect that the findings of this study would have been similar if any other Germanic language (including English) was used.

The contribution of this paper is a thorough and fair comparison of the involved preprocessing techniques. There have been earlier studies of hypernym extraction with either lexical or dependency extraction patterns. However, these studies applied the techniques to a variety of different data sets and used different evaluation techniques. We will apply the two methods to the same data, evaluate the results in a consistent manner and examine the differences.

After this introduction, we will describe the task, the preprocessing methods and the evaluation setting in more detail. In the third section, we will show how our experiments were set up and present the results. Section four contains a detailed discussion of the two methods and their effect on the extraction task. In the final section of the paper, we will present some concluding remarks.

2 Task and methods

We will apply two different preprocessing methods to the task of extracting lexical information from text. In the next sections we describe this task, discuss different methods for preprocessing the data and outline the method used for evaluating the results.

2.1 Extracting hypernym relations

We will concentrate on extracting a single type of lexical relation: hypernymy. Word A is a hypernym of word B if the meaning of A both covers the meaning of B and is broader. For example, *color* is a hypernym of *red* which in turn is a hypernym of *scarlet*. If A is a hypernym of B than B is a hyponym of A.

There has been quite a lot of work on extracting hypernymy pairs from text. The pioneering work of Hearst (1992) applied fixed patterns like NP_1 , *especially* NP_2 to derive that NP_1 is a hypernym of NP_2. Lately there has been a lot of interest in acquiring such text patterns using a set of hypernymy examples, e.g. Pantel et al. (2004) and Snow (2006). Application of such techniques has not been restricted to English but also involved other languages such as Dutch (Tjong Kim Sang and Hofmann, 2007). Recent work has also examined extracting hypernym information from structured data, like Wikipedia (Sumida and Torisawa, 2008).

For our extraction work, we will closely follow the approach described in Snow et al. (2006):

1. Collect from a text corpus phrases (consecutive word sequences from a single sentence) that contain a pair of nouns
2. Mark each phrase as containing a hypernym pair or a non-hypernym pair according to a lexical resource
3. Remove the noun pair from the phrases and register how often each phrase is associated by hypernym pairs and by non-hypernym pairs
4. Use this information for training a machine learning system to predict whether two nouns are a hypernym-hyponym pair based on the phrases in which they occur in a text corpus

For example, we find two phrases: *colors such as cyan* and *colors such as birds*, both of which contain the basic phrase *such as*. We mark the first phrase as a hypernym phrase (*color* is a hypernym of *cyan*) while the second is marked as non-hypernym (*color* is not a hypernym of *bird*). Thus the pattern *such as* will receive a positive point and a negative point. A machine learning algorithm can deduce from these numbers that two other nouns occurring in the same pattern will have an estimated probability of 50% of being related according to hypernymy. The learner can use information from other patterns to obtain a better estimation of this probability.

2.2 Lexical patterns

We use two different text preprocessing methods which automatically assign linguistic information to sentences. The first preprocessing method has the

advantage of offering a fast analysis of the data but its results are less elaborate than those of the second method. The first method consists of three steps:

- Tokenization: sentence boundaries are detected and punctuation signs are separated from words
- Part-of-speech tagging: part-of-speech classes like noun and verb are assigned to words
- Lemmatization: words are reduced to their basic form (lemma)

The analysis process would convert a phrase like *Large cities in northern England such as Liverpool are beyond revival.* to lemmas and their associated part-of-speech tags: *large*/JJ *city*/NN *in*/IN *north*/JJ *England*/NNP *such*/DT *as*/IN *Liverpool*/NNP *be*/VB *beyond*/IN *revival*/NN ./.

Like in the work of Snow et al. (2005), the target phrases for hypernym extraction are two noun phrases, with a maximum of three tokens in between and one or two optional extra tokens (a non-head token of the first noun phrase and/or one of the second noun phrase). The lexical preprocessing method uses two basic regular expressions for finding noun phrases: *Determiner? Adjective* Noun+* and *ProperNoun+*. It assumes that the final token of the matched phrase is the head. Here is one set of four patterns which can be derived from the example sentence:

1. *NP in NP*
2. *large NP in NP*
3. *NP in north NP*
4. *large NP in north NP*

The patterns contain the lemmas rather than the words of the sentence in order to allow for general patterns. For the same reason, the noun phrases have been replaced by the token NP. Each of the four patterns will be used as evidence for a possible hypernymy relation between the two noun phrase heads *city* and *England*. As a novel extension to the work of Snow et al., we included two additional variants of each pattern in which either the first NP or the second NP was replaced by its head:

5. *city in NP*
6. *NP in England*

This enabled us to identify among others appositions as patterns: *president NP*.

2.3 Dependency patterns

A dependency analysis contains the same three steps used for finding lexical patterns: tokenization, part-of-speech tagging and lemmatization. Additionally, it includes a fourth step:

- Dependency parsing: find the syntactic dependency relations between the words in each sentence

The syntactic analysis is head-based which means that for each word in the sentence it finds another word that dominates it. Here is a possible analysis of the previous example sentence:

large:JJ:MOD:NN:*city*
city:NN:SUBJ:VBD:*be*
in:IN:MOD:NN:*city*
north:JJ:MOD:NNP:*England*
England:NNP:OBJ1:IN:*in*
such:DT:MOD:IN:*as*
as:IN:MOD:NN:*city*
Liverpool:NNP:OBJ1:IN:*as*
be:VB:–:–:–
beyond:IN:MOD:VB:*be*
revival:NN::OBJ1:IN:*beyond*

Each line contains a lemma, its part-of-speech tag, the relation between the word and its head, the part-of-speech tag of its head and the lemma of the head word. Our work with dependency patterns closely follows the work of Snow et al. (2005). Patterns are defined as dependency paths with at most three intermediate nodes between the two focus nouns. Additional satellite nodes can be present next to the two nouns. The dependency patterns contain more information than the lexical patterns. Here is one of the patterns that can be derived for the two noun phrases *large cities* and *northern England* in the example sentence:

NP_1:NN:SUBJ:VBD:
in:IN:MOD:NN:NP_1
NP_2:NNP:OBJ1:IN:*in*

The pattern defines a path from the head lemma *city* via *in*, to *England*. Note that lemma information linking outside this pattern (*be* at the end of the first line) has been removed and that lemma information

from the target noun phrases has been replaced by the name of the noun phrase (NP_1 at the end of the second line). For each dependency pattern, we build six variants similar to the six variants of the lexical patterns: four with additional information from the two noun phrases and two more with head information of one of the two target NPs.

Both preprocessing methods can identify phrases like *N such as* N_1 *,* N_2 *and* N_3 as well. Such phrases produce evidence for each of the pairs (N,N_1), (N,N_2) and (N,N_3). These three noun pairs will be included in the data collected for the patterns that can be derived from the phrase.

We expect that an important advantage of using dependency patterns over lexical patterns will be that the former offer a wider coverage. In the example sentence, no lexical pattern will associate *city* with *Liverpool* because there are too many words in between. However, a dependency pattern will create a link between these two words, via the word *as*. This will enable the dependency patterns to find out that *city* is a hypernym of *Liverpool*, where the lexical patterns are not able to do this based on the available information.

The two preprocessing methods generate a large number of noun pairs associated by patterns. Like Snow et al. (2005), we keep only noun pairs which are associated by at least five different patterns. The same constraint is enforced on the extraction patterns: we keep only the patterns which are associated by at least five different noun pairs. The data is converted to binary feature vectors representing noun pairs. These are training data for a Bayesian Logistic Regression system, BBRtrain (Genkin et al., 2004). We use the default settings of the learning system and test its prediction capability in a binary classification task: whether two nouns are related according to hypernymy or not. Evaluation is performed by 10-fold cross validation.

2.4 Evaluation

For parameter optimization we need an automatic evaluation procedure, since repeated manual checks of results generated by different versions of the learner require too much time. We have adopted the evaluation method of Snow et al (2006): compare the generated hypernyms with hypernyms present in a lexical resource, in our case the Dutch part of EuroWordNet (1998).

This choice results in two restrictions. First, we will only consider pairs of known words (words that are present in the lexical resource) for evaluation. We have no information about other words so we make no assumptions about them. Second, if two words appear in the lexical resource but not in the hypernym relation of that same resource then we will assume that they are unrelated. In other words, we assume the hypernymy relation specified in the lexical resource as complete (like in the work of Snow et al. (2006)).

We use standard evaluation scores. We will compute precision and recall for the candidate hypernyms, as well as the related $F_{\beta=1}$ rate, the harmonic mean between precision and recall. Precision will be computed against all chosen candidate hypernyms. However, recall will only be computed against the positive noun pairs which occur in the phrases selected by the examined method. The different preprocessing methods may cause different numbers of positive pairs to be selected. Only these pairs will be used for computing recall scores. Others will be ignored. For this reason we will report the selected number of positive target pairs in the result tables as well[1].

3 Experiments and results

We have applied the extraction techniques to two different Dutch corpora. The first is a collection of texts from the news domain. It consists of texts from five different Dutch news papers from the Twente News Corpus collection. Two versions of this corpus exist. We have worked with the version which contains the years 1997-2005 (26 million sentences and 450 million tokens). The second corpus is the Dutch Wikipedia. Here we used a version of October 2006 (5 million sentences and 58 million words).

Syntactic preprocessing of the material was done with the Alpino parser, the best available parser for Dutch with a labeled dependency accuracy of 89% (Van Noord, 2006). Rather than performing the parsing task ourselves, we have relied on an available parsed treebank which included the text corpora

[1] In a seperate study we have shown that the observed differences between the two methods remain the same when recall is computed over sets of similar sizes (Tjong Kim Sang, 2009).

that we wanted to use (Van Noord, 2009).

The parser also performs part-of-speech tagging and lemmatization, tasks which are useful for the lexical preprocessing methods. However, taking future real-time applications in mind, we did not want the lexical processing to be dependent on the parser. Therefore we have developed an in-house part-of-speech tagger and lemmatizer based on the material created in the Corpus Spoken Dutch project (Eynde, 2005). The tagger achieved an accuracy of 96% on test data from the same project while the lemmatizer achieved 98%.

We used the Dutch part of EuroWordNet (Vossen, 1998) as the gold standard lexical resource, both for training and testing. In the lexicon, many nouns have different senses. This can cause problems for the pattern extraction process. For example, if a noun N_1 with sense X is related to another noun N_2 then the appearance of N_1 with sense Y with N_2 in the text may be completely accidental and say nothing about the relation between the two words. In that case it would be wrong to regard the context of the two words as an interesting extraction pattern.

There are several ways to deal with this problem. One is to automatically assign senses to words. However we do not have a reliable sense tagger for Dutch at our disposal. Another method was proposed by Snow et al (2005): assume that every word bears its most frequent sense. But this is also information which we lack for Dutch: our lexical resource does not contain frequency information for word senses. We have chosen the approach suggested by Hofmann and Tjong Kim Sang (2007): remove all nouns with multiple senses from the data set and use only the monosemous words for finding good extraction patterns. This restriction is only imposed in the training phase. We consider both monosemous words and polysemous words in the evaluation process.

We imposed two additional restrictions on the lexical resource. First, we removed the top noun of the hypernymy hierarchy (*iets*) from the list of valid hypernyms. This word is a valid hypernym of any other noun. It is not an interesting suggestion for the extraction procedure to put forward. Second, we restricted the extraction procedure to propose only known hypernyms as candidate hypernyms. Nouns that appeared in the lexical resources only as hyponyms (leaf nodes of the hypernymy tree) were never proposed as candidate hypernyms. This made sense for our evaluation procedure which is only aimed at finding known hypernym-hyponym pairs.

lexical patterns

Data source	Targ.	Prec.	Recall	$F_{\beta=1}$
AD	620	55.8%	27.9%	37.2
NRC	882	50.4%	23.8%	32.3
Parool	462	51.8%	21.9%	30.8
Trouw	607	54.1%	25.9%	35.0
Volkskrant	970	49.7%	24.1%	32.5
Newspapers	3307	43.1%	26.7%	33.0
Wikipedia	1288	63.4%	44.3%	52.1

dependency patterns

Data source	Targ.	Prec.	Recall	$F_{\beta=1}$
AD	706	42.9%	30.2%	35.4
NRC	1224	26.2%	25.3%	25.7
Parool	584	31.2%	23.8%	27.0
Trouw	760	35.3%	29.0%	31.8
Volkskrant	1204	29.2%	25.5%	27.2
Newspapers	3806	20.7%	29.1%	24.2
Wikipedia	1580	61.9%	47.0%	53.4

Table 1: Hypernym extraction scores for the five newspapers in the Twente News Corpus (AD, NRC, Parool, Trouw and Volkskrant) and for the Dutch Wikipedia. The Targets column shows the number of unique positive word pairs in each data set. The Dutch Wikipedia contains about as much data as one of the newspaper sections.

We performed two hypernym extraction experiments, one which used lexical extraction patterns and one which used dependency patterns[2]. The results from the experiments can be found in Table 1. The newspaper F-scores obtained with lexical patterns are similar to those reported for English (Snow et al., 2005, 32.0) but the dependency patterns perform worse. Both approaches perform well on Wikipedia data, most likely because of the more repeated sentence structures and the presence of many definition sentences. For newspaper data, lexical patterns outperform dependency patterns both for precision and $F_{\beta=1}$. For Wikipedia data the differences are smaller and in fact the dependency pat-

[2]The software used in these experiment has been made available at http://www.let.rug.nl/erikt/cornetto/D08.zip

terns obtain the best F-score. For all data sets, the dependency patterns suggest more related pairs than the lexical patterns (column Targets). The differences between the two pattern types are significant ($p < 0.05$) for all evaluation measures for Newspapers and for positive targets and recall for Wikipedia.

4 Result analysis

In this section, we take a closer look at the results described in the previous section. We start with looking for an explanation for the differences between the scores obtained with lexical patterns and dependency patterns. First we examine the results for Wikipedia data and then the results for newspaper data. Finally, we perform an error analysis to find out the strengths and weaknesses of each of the two methods.

4.1 Wikipedia data

The most important difference between the two pattern types for Wikipedia data is the number of positive targets (Table 1). Dependency patterns find 23% more related pairs in the Wikipedia data than lexical patterns (1580 vs. 1288). This effect can also be simulated by changing the size of the corpus. If we restrict the data set of the dependency patterns to 70% of its current size then the patterns retrieve a similar number of positive targets as the lexical patterns, 1289, with comparable precision, recall and $F_{\beta=1}$ scores (62.5%, 46.6% and 53.4). So we expect that the effect of applying the dependency patterns is the same as applying the lexical patterns to 43% more data.

4.2 Newspaper data

Performance-wise there seems to be only a small difference between the two preprocessing methods when applied to the Wikipedia data set. However, when we examine the scores obtained on the newspaper data (Table 1) then we find larger differences. Dependency patterns remain finding more positive targets and obtaining a larger recall score but their precision score is disappointing. However, when we examine the precision-recall plots of the two methods (Figure 1, obtained by varying the acceptance threshold of the machine learner), they are almost indistinguishable. The performance line for lexical patterns extends further to the left than the one of

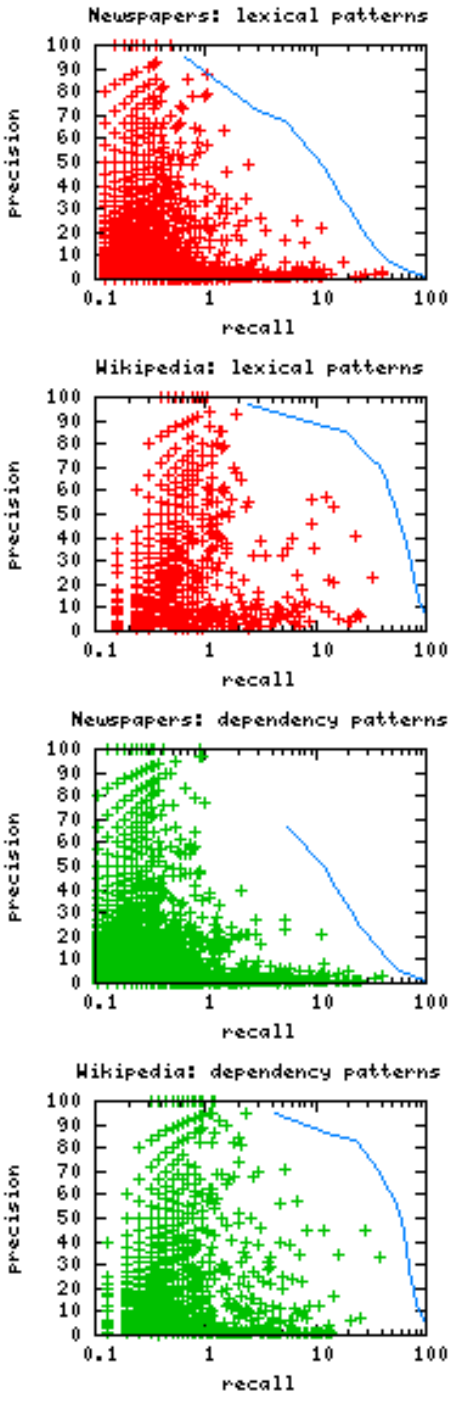

Figure 1: Performance of individual hypernym extraction patterns applied to the combination of five newspapers and Wikipedia. Each + in the graphs represent a different extraction pattern. The precision-recall graphs for the machine learner (lines) are identical for each data source except for the extended part of the performance line for lexical patterns.

lexical patterns applied to Newspapers

Key Phrase	Targ.	Prec.	Recall	$F_{\beta=1}$
N and other N	376	22.0%	11.4%	15.0
N such as N	222	25.1%	6.7%	10.6
N like N	579	7.6%	17.5%	10.6
N , such as N	263	15.6%	8.0%	10.5
N (N	323	7.5%	9.8%	8.5

dependency patterns applied to Newspapers

Key Phrase	Targ.	Prec.	Recall	$F_{\beta=1}$
N and other N	420	21.1%	11.0%	14.5
N be a N	451	8.2%	11.8%	9.7
N like N	205	27.3%	5.4%	9.0
N be N	766	5.7%	20.1%	8.8
N such as N	199	22.4%	5.2%	8.5

lexical patterns applied to Wikipedia

Key Phrase	Targ.	Prec.	Recall	$F_{\beta=1}$
N be a N	294	40.8%	22.8%	29.3
N be N	418	22.9%	32.5%	26.9
a N be N	185	53.3%	14.4%	22.6
N such as N	161	57.5%	12.5%	20.5
N (N	188	21.2%	14.6%	17.3

dependency patterns applied to Wikipedia

Key Phrase	Targ.	Prec.	Recall	$F_{\beta=1}$
N be N	609	33.6%	38.5%	35.9
N be a N	452	44.3%	28.6%	34.8
the N be N	258	34.0%	16.3%	22.1
a N be N	184	44.7%	11.6%	18.5
N N	234	16.6%	14.8%	15.6

Table 2: Best performing extraction patterns according to F-scores.

the dependency patterns but the remainder of the two graphs overlap. The measured performances in Table 1 are different because the machine learner put the acceptance level for extracted pairs at different points of the graph: the performance lines in both newspaper graphs contain (recall,precision) points (26.7%,43.1%) and (29.1%,20.7%).

We are unable to find major differences in the results of the two approaches. We conclude that, apart from an effect which can be simulated with some extra data, there is no difference between preprocessing text with shallow methods and with a full dependency parser.

56	—	covered by other patterns
12	48%	required full parsing
6	24%	lemmatization errors
3	12%	omitted for lack of support
3	12%	pos tagging errors
1	4%	extraction pattern error
81	100%	

45	—	covered by other patterns
38	64%	parsing errors
10	17%	lemmatization errors
7	12%	extraction pattern errors
3	5%	omitted for lack of support
1	2%	pos tagging error
104	100%	

Table 3: Primary causes of recall errors made by the lexical pattern *N such as N* (top) and the best performing corresponding dependency pattern (bottom).

4.3 Error analysis

Despite the lack of performance differences between the two preprocessing methods, there are still internal differences which cause one method to generate different related word pairs than the other. We will now examine in detail two extraction patterns and specify their distinct effects on the output results. We hope that by carefully examining their output we can learn about the strengths and weaknesses of the two approaches.

We take a closer look at extraction pattern *N such as N* for Newspaper data (second best for lexical patterns and fifth best for dependency patterns, see Table 2). The lexical pattern found 222 related word pairs while the dependency pattern discovered 199. 118 of these pairs were found by both patterns which means that the lexical pattern missed 81 of the pairs while the dependency pattern missed 104.

An overview of the cause of the recall errors can be found in Table 3. The two extraction patterns do not overlap completely. The dependency parser ignored punctuation signs and therefore the dependency pattern covers both phrases with and without punctuation. However, these phrase variants result in different lexical patterns. This is the cause for 56 hypernyms being missed by the lexical pattern.

Meanwhile there is a difference between a dependency pattern without the conjunction *and* and one with the conjunction, while there is a unified lexical pattern processing both phrases with and without conjunctions. This caused the dependency pattern to miss 45 hypernyms. However, all of these 'missed' hypernyms are handled by other patterns.

The main cause of the recall differences between the two extraction patterns was the parser. The dependency pattern found twelve hypernyms which the lexical pattern missed because they required an analysis which was beyond part-of-speech tagging and the basic noun phrase identifier used by the lexical preprocessor. Six hypernyms required extending a noun phrase with a prepositional phrase, five needed noun phrase extension with a relative clause and one involved appositions. An example of such a phrase is *illnesses caused by vitamin deficits, like scurvy and beriberi*.

However, the syntactic information that was available to the dependency pattern did also have a negative effect on its recall. 38 of the hypernyms detected by the lexical pattern were missed by the dependency pattern because there was a parsing error in the relevant phrase. In more than half of the cases, this involved attaching the phrase starting with *such as* at an incorrect position. We found that a phrase like N_1 *such as* N_2 *,* N_3 *and* N_4 could have been split at any position. We even found some cases of prepositional phrases and relative clauses incorrectly being moved from other positions in the sentence into the target phrase.

Other recall error causes appear less frequently. The two preprocessing methods used different lemmatization algorithms which also made different errors. The effects of this were visible in the errors made by the two patterns. Some hypernyms that were found by both patterns but were not present in both results because of insufficient support from other patterns (candidate hypernyms should be supported by at least five different patterns). The effect of errors in part-of-speech tags was small. Our data analysis also revealed some inconsistencies in the extraction patterns which should be examined.

5 Concluding remarks

We have evaluated the effects of two different preprocessing methods for a natural language processing task: automatically identifying hypernymy information. The first method used lexical patterns and relied on shallow processing techniques like part-of-speech tagging and lemmatization. The second method used dependency patterns which relied on additional information obtained from dependency parsing.

In earlier work, McCarthy et al. (2007) found that for word sense disambiguation using thesauri generated from dependency relations perform only slightly better than thesauri generated from proximity-based relations. Jijkoun et al. (2004) showed that information obtained from dependency patterns significantly improved the performance of a question answering system. Li and Roth (2001) report that preprocessing by shallow parsing allows for a more accurate post-processing of ill-formed sentences than preprocessing with full parsing.

Our study supports the findings of McCarthy et al. (2007). We found only minor differences in performances between the two preprocessing methods. The most important difference: about 20% extra positive cases that were identified by the dependency patterns applied to Wikipedia data, can be overcome by increasing the data set of the lexical patterns by half. We believe that obtaining more data may often be easier than dealing with the extra computing time required for parsing the data. For example, in the course of writing this paper, we had to refrain from using a recent version of Wikipedia because parsing the data would have taken 296 *days* on a single processor machine compared with a single hour for tagging the data.

References

Eugene Charniak. 1999. A maximum-entropy inspired parser. Technical Report CS-99-12, Brown University.

Frank Van Eynde. 2005. *Part of Speech Tagging en Lemmatisering van het Corpus Gesproken Nederlands*. K.U. Leuven. (in Dutch).

Alexander Genkin, David D. Lewis, and David Madigan. 2004. *Large-Scale Bayesian Logistic Regression for Text Categorization*. Technical report, Rutgers University, New Jersey.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of ACL-92*. Newark, Delaware, USA.

Katja Hofmann and Erik Tjong Kim Sang. 2007. Automatic extension of non-english wordnets. In *Proceedings of SIGIR'07*. Amsterdam, The Netherlands (poster).

Valentin Jijkoun, Maarten de Rijke, and Jori Mur. 2004. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of Coling'04*. Geneva, Switzerland.

Xin Li and Dan Roth. 2001. Exploring evidence for shallow parsing.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Caroll. 2007. Unsupervised acquisition of predominant word senses. *Computational Linguistics*, 33(4).

Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. 2004. Towards terascale knowledge acquisition. In *Proceedings of COLING 2004*, pages 771–777. Geneva, Switzerland.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS 2005*. Vancouver, Canada.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING/ACL 2006*. Sydney, Australia.

Asuka Sumida and Kentaro Torisawa. 2008. Hacking wikipedia for hyponymy relation acquisition. In *Proceedings of IJCNLP 2008*. Hyderabad, India.

Mihai Surdeanu, Richard Johansson, Lluís Màrquez, Adam Meyers, and Joakim Nivre. 2008. The conll-2008 shared task on joint learning of syntactic and semantic dependencies. In *Proceedings of CoNLL-2008*. Manchester, UK.

Erik Tjong Kim Sang and Katja Hofmann. 2007. Automatic extraction of dutch hypernym-hyponym pairs. In *Proceedings of CLIN-2006*. Leuven, Belgium.

Erik Tjong Kim Sang. 2009. To use a treebank or not – which is better for hypernym extraction. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT 7)*. Groningen, The Netherlands.

Gertjan Van Noord. 2006. At last parsing is now operational. In Piet Mertens, Cedrick Fairon, Anne Dister, and Patrick Watrin, editors, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*.

Gertjan Van Noord. 2009. Huge parsed corpora in lassy. In *Proceedings of TLT7*. LOT, Groningen, The Netherlands.

Piek Vossen. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publisher.