Unsupervised Lexical Acquisition: Proceedings of the Workshop of the
ACL Special Interest Group on the Lexicon (SIGLEX), Philadelphia,
July 2002, pp. 26-33. Association for Computational Linguistics.

Building a hyponymy lexicon with hierarchical structure

Sara Rydin
rydin@speech.kth.se
Centre for Speech Technology (CTT)
KTH Stockholm, Sweden
GSLT

Abstract

Many lexical semantic relations, such as the hyponymy relation, can be extracted from text as they occur in detectable syntactic constructions. This paper shows how a hypernym-hyponym based lexicon for Swedish can be created directly from a news paper corpus. An algorithm is presented for building partial hierarchical structures from non domain-specific texts.

1 Introduction

Automatic acquisition of information on semantic relations from text has become more and more popular during the last ten to fifteen years. The goal has been to build various types of semantic lexicons for use in natural language processing (NLP) systems, such as systems for information extraction/retrieval or dialog systems. The lexicons are used to introduce extended semantic knowledge into the different systems.

Hand-built general-purpose lexicons, such as the WordNet (Fellbaum, 1998), have often been used to bring semantic knowledge into NLP-systems. Two important problems concerning (semantic) lexicons are those of domain coverage and updates. Firstly, a general-purpose lexicon cannot be expected to cover all specific words used in different sub-domains. Therefore, the need for domain-specific lexicons has recently been brought to the surface.

Secondly, any lexicon, general or specific, has to be updated from time to time, in order to keep up with new words and new uses of existing words. Our minimally supervised method for automatically building partial hierarchies presents one way to solve the update problem.

The objective of this project is to automatically build a hierarchical hyponymy lexicon of noun phrases given large, part-of-speech tagged and lemmatized corpora that are not restricted to one specific domain or topic. The lexicon will thus, reflect partial hierarchical hyponymy structures that bring forward extended hypernym-hyponym relations.

Section 2 describes previous work in the area of automatic acquisition of semantic lexicons, section 3 elaborates on the principles for this work, and the remaining sections describe the implementation as well as the evaluation of the algorithm for building a hierarchical hyponymy lexicon.

2 Previous work

One of the first studies on acquisition of hyponymy relations was made by Hearst (1992). She found that certain lexico-syntactic constructions can be used as indicators of the hyponymy relation between words in text. Example 1 shows a relation of this kind and an example. The noun phrase 'NP_h' is a hypernym and '$((NP_2,) * NP_n\, and|or)\, NP_1$' is one or more (conjoined) noun phrases that are the hyponyms:

$$\textbf{such}\ NP_h\ \textbf{as}\ ((NP_2,) * NP_n\, and|or)\, NP_1 \quad (1)$$

'such cars as Volvo, Seat and Ford'

Hearst proposed furthermore, that new syntactic patterns can be found in the following way:

1. Use the list of hypernyms-hyponyms found by the type of pattern described above to search for places in the corpus where the two expressions occur syntactically close to each other. Save the syntactic examples.

2. Examine all saved syntactic environments and find new useful syntactic patterns.

3. Use each new pattern to find more hypernym-hyponym examples. Continue at 1.

Caraballo (1999) uses a hierarchical clustering technique to build a hyponymy hierarchy of nouns. The internal nodes are labeled by the syntactic constructions from Hearst (1992). Each internal node in the hierarchy can be represented by up to three nouns.

Work by Riloff & Shepherd (1997) and Charniak & Roark (1998) aims to build semantic lexicons where the words included in each category or entry are related to, or are a member of the category.

Sanderson & Croft (1999) build hierarchical structures of concepts on the basis of generality and specificity. They use material divided by different text categories and base the decision of subsumption on term co-occurrence in the different categories. A term *x* is said to subsume *y* if the documents in which *y* occurs are a subset of the documents in which *x* occurs. The relations between concepts in their subsumption hierarchy are of different kinds (among other the hyponymy relation), and are unlabeled.

The work most similar to ours is that of Morin & Jacquemin (1999). They produce partial hyponymy hierarchies guided by transitivity in the relation. But while they work on a domain-specific corpus, we will acquire hyponymy data from a corpus which is not restricted to one domain.

3 Principles for building a hierarchical lexicon

This section will describe the principles behind our method for building the hierarchical structures in a lexicon.

As the objective is to build a nominal hyponymy lexicon with partial hierarchical structures, there are conditions that the hierarchical structures should meet. The structures can each be seen as separate hyponymy hierarchies, and for each hierarchy the following criteria should be fulfilled:

1. A hierarchy has to be strict, so that every child node in it can have one parent node only.

2. The words or phrases forming the nodes in a hierarchy should be disambiguated.

3. The organization in a hierarchy should be such that every child node is a hyponym (i.e. a type/kind) of its parent.

Generally, principle 1-2 above are meant to prevent the hierarchies from containing ambiguity. The built-in ambiguity in the hyponymy hierarchy presented in (Caraballo, 1999) is primarily an effect of the fact that all information is composed into *one* tree. Part of the ambiguity could have been solved if the requirement of building one tree had been relaxed.

Principle 2, regarding keeping the hierarchy ambiguity-free, is especially important, as we are working with acquisition from a corpus that is not domain restricted. We will have to constrain the way in which the hierarchy is growing in order to keep it unambiguous. Had we worked with domain-specific data (see e.g. Morin and Jaquemin (1999)), it would have been possible to assume only one sense per word or phrase.

The problem of building a hyponymy lexicon can be seen as a type of classification problem. In this specific classification task, the hypernym is the class, the hyponyms are the class-members, and classifying a word means connecting it to its correct hypernym. The algorithm for classification and for building hierarchies will be further described in section 6.

4 Corpus and relevant terms

This work has been implemented for Swedish, a Germanic language. Swedish has frequent and productive compounding, and morphology is richer compared to, for example, English. Compounding affects the building of any lexical resource in that the number of different word types in the language is larger, and thus, the problems of data sparseness become more noticeable. In order to, at least partly,

overcome the data sparseness problem, lemmatization has been performed. However, no attempt has been made to make a deeper analysis of compounds.

The corpus used for this research consists of 293,692 articles from the Swedish daily news paper 'Dagens Nyheter'. The corpus was tokenized, tagged and lemmatized. The tagger we used, implemented by Megyesi (2001) for Swedish, is the TnT-tagger (Brants, 2000), trained on the SUC Corpus (Ejerhed et al., 1992). After preprocessing, the corpus was labeled for base noun phrases (baseNP). A baseNP includes optional determiners and/or premodifiers, followed by nominal heads.

Naturally, conceptually relevant terms, rather than noun phrases, should be placed in the lexicon and the hierarchies. For reasons of simplification, though, the choice was made as to treat nominal heads with premodifying nouns in genitive (within the limits of the baseNP described above) as the relevant terms to include in the hierarchies. However, premodifiers describing amounts, such as 'kilo', are never included in the relevant terms.

5 Lexico-syntactic constructions

Lexico-syntactic constructions are extracted from the corpus, in the fashion suggested by Hearst (1992). Five different Swedish constructions has been chosen – constructions 2-6 below – as a basis for building the lexicon (an example with the English translation is given below for each construction)[1]:

$$sådana\ NP_h\ som\ ((NP_n,\ )*\ NP_2\ och|eller)\ NP_1 \quad (2)$$

'sådana känslor som medkänsla och barmhärtighet' /lit. such feelings as sympathy and compassion/

$$NP_h\ (så)\ som\ ((NP_n,\ )*\ NP_2\ och|eller)\ NP_1 \quad (3)$$

'exotiska frukter som papaya, pepino och mango' /lit. exotic fruits such as papaya, pepino and mango/

$$NP_1\ (,\ NP_n)*\ och|eller\ annan\ NP_h \quad (4)$$

'trafikinformation och annan information' /lit. information on traffic and other information/

$$NP_1\ (,\ NP_n)*\ och|eller\ liknande\ NP_h \quad (5)$$

[1]Construction six requires a numerical expression (num.expr.) greater than one.

'riksdagen, stadsfullmäktige och liknande församlingar' /lit. the Swedish Parliament, the town councilor and similar assemblies/

$$NP_1\ (,\ NP_n)*\ och|ellerNP_2,\ num.expr.NP_h \quad (6)$$

'Österleden och Västerleden, de två motorvägsprojekt' /lit. the East way and the West way, the two highway projects/

The basic assumption is that these constructions (henceforth called hh-constructions), yield pairs of terms between which the hyponymy relation holds. After a manual inspection of 20% of the total number of hh-constructions, it was estimated that 92% of the hh-constructions give us correct hyponymy relations. Erroneous hh-constructions are mainly due to problems with, for example, incorrect tagging, but also change in meaning due to PP-attachment.

6 Building the hierarchical lexicon

To give an accurate description of the algorithm for building the lexicon, the description here is divided into several parts. The first part describes how hypernyms/hyponyms are grouped into classes, building an unambiguous lexicon base. The second part describes how arrangement into hierarchical structures is performed from this unambiguous data. Last, we will describe how the lexicon is extended.

6.1 Classification

There are two straightforward methods that can be used to classify the data from the hh-constructions. The first would be to group all hypernyms of the same lemma into one class. The second would be to let each hypernym token (independently of their lemma) initially build their own class, and then try to group tokens according to their sense. The first method is suitable for building classes from the hh-constructions for a domain-specific corpus. However, when working with a news paper corpus, as in our case, this method would lead to possible ambiguity in the classes, as hypernyms of the same lemma can have more than one sense.

Thus, we choose to take the second, more cumbersome approach in order to avoid all possible ambigu-

'...fever, pain, and other symptoms...'
Hypernym: *symptom*; Hyponyms: *fever*, *pain*
→
class: symptom_3
class feature: *symptom*; class members: *fever*, *pain*

Table 1: Example of how an initial class is created from a simple hh-construction.

ity in the lexicon. Avoiding ambiguity is important as the result of classification will be used as a base for building a lexicon with hierarchical structures.

Initially, the hypernym and hyponyms of the hh-constructions from the text are used to build a base for a class system. An example of how an initial class is created from a simple hh-construction is given in Table 1. Each class *X_N* has a class feature *X* which is the hypernym's lemma, where *N* is a unique number designating the unique class and where the class members are the hyponym lemmas.

After this initial step, the unique classes are grouped into larger classes. Constraints are put on the grouping process in order to keep the classes unambiguous. [2] Two classes A and B can only be collapsed if they fulfill the following two prerequisites:

1. The class features of the classes have to be the same.

2. There has to be a non-empty intersection in class members between the classes.

An example of a collapsing operation of this kind is given in Table 2. As can be seen in the table, the method captures correct sense distinctions as well as incorrect ones (i.e. two classes are created when there should be only one). The effect of this will be further discussed in section 8. Note however, that some incorrect sense distinctions introduced here are corrected through the introduction of hierarchical structure, which will be discussed in the next section.

6.2 Building hierarchical structure

Hierarchical structure is introduced in the lexicon through a number of rules, which are directed by

[2] Also, system internally, all words, hypernyms and hyponyms have unique index number attached to them.

symptom_1: $symptom_1 - killing, robbery$
symptom_2: $symptom_2 - fever_1, ailment$
symptom_3: $symptom_3 - fever_2, pain$
symptom_4: $symptom_4 - eczema, irritation$
→
symptom_1: $symptom_1 - killing, robbery$
symptom_2: $symptom_2 - fever, ailment, pain$
symptom_4: $symptom_4 - eczema, irritation$

Table 2: Four classes are collapsed into three classes. After the collapse, correct sense distinction is kept between the class denoted 1 and the classes 2 and 4. Incorrect sense distinction is created between the classes denoted 2 and 4.

the over all principle of transitivity in the hyponymy relation. That is, if *X* is a kind of *Y*, and *Y* is a kind of *Z*, then the two classes containing these pairs can only be composed if the hyponymy relation also holds between *X* and *Z*. In practice, the three hypernym-hyponym pairs *X-Y*, *Y-Z* and *X-Z* all have to be found in our corpus.[3]

Next, we will turn to the outline of the implementation for building the hierarchical structures from the classes created through the method described in the previous section:

1. For each class k among all classes:
   a. find all sets of classes that can be used in building hierarchies with class k.
   b. choose one set of classes that should be used.

2. Compose all chosen sets of classes.

3. Build trees that reflect all the implemented compositions.

A typical hierarchical structure of the kind that is built here can be seen in Figure 1. The algorithm for building this hierarchical structure will now be described in more detail:

Searching for a set of classes for composition is performed according to the transitivity principle described above. For each hypernym-hyponym pair $X_1 - Y_1$ (see example below), search for two other classes (class *2* and *3*) that contains the hypernym-hyponym pairs $X_2 - Z_1$ and $Y_2 - Z_2$

[3] For further discussion about transitivity in trees, see (Woods, 1997).

respectively:

Class 1 : $X_1 - ...Y_1, ...$
/symptom_2: $symptom_2 - fever, ailment_1, pain$/
Class 2 : $X_2 - ...Z_1, ...$
/symptom_5: $symptom_5 - infection_1, headache$/
Class 3 : $Y_2 - ...Z_2, ...$
/ailment_1: $ailment_2 - infection_2, fatigue$/

For each class, all sets of classes that can be used for compositions into hierarchies ('classes *2 + 3*' constitutes one set for class *1*) are stored. From these sets of classes, one is randomly chosen for implementation.[4]

To implement the composition of the classes into a hierarchy we:

1. Connect Z_1 and Z_2, i.e. remove Z_1
   /symptom_5: $symptom_5$ - $headache$/

2. Connect Y_1 and Y_2, i.e. rename Y_1 by Y_2
   /symptom_2: $symptom_2$ - $fever$, $ailment_2$, $pain$/

3. Move all remaining class members in class 2 to class 1 and remove class 2
   /symptom_2: $symptom_2$ - $fever$, $ailment_1$, $pain$, $headache$/

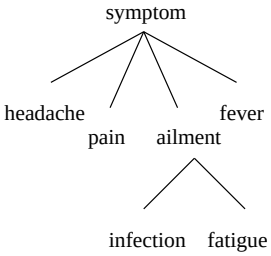

Figure 1: Hierarchical structure.

It is also possible to compose sets of classes where class 1 = class 2, but in that case, step 3. is left out. The result is, in any case, two modified classes, where the classes are linked through the term Y_2. In cases where class 1 $\neq$ class 2, class 2 is erased and its class members are placed as class members in class 3. The result of composing class 1-3 is:

Class 1 : $X_1 - ...Y_2, ...$
/symptom_2: $symptom$ - $fever$, $ailment_1$, $pain$, $headache$/
Class 3 : $Y_2 - ...Z_2, ...$
/ailment_1: $ailment_1$ - $infection$, $fatigue$/

In the final trees, all different compositions are reflected. In this way, several compositions might cowork to build trees, and the more compositions that are used, the deeper the tree will be.

It is worth noting that, when any tree is built, end nodes (i.e. non-internal hyponyms) with the same lemma as other end nodes in the collapsed tree are moved downwards in the tree. The goal is to keep only one instance, i.e. the one that is placed lowest in the tree. Also, measures are taken all along in the building process in order to keep the tree acyclic.

6.3 Extending the lexicon

Obviously, apart from the hypernym-hyponym data that we get from the hh-constructions listed in section 5, more data can be found in text. In order to capture some of the data, we propose a similar but simpler algorithm to that of Hearst (1992). The algorithm is simpler in that it does not search for new syntactic environments that reveals the hypernym-hyponym relation. Instead, it relies on the general syntactic pattern in step 2 (below) for finding new lexical hypernym-hyponym data:

1. Look through the previously extracted hh-constructions and extract the pairs of hypernyms-hyponyms where the frequency of the pair is higher than 2.

2. Search in new data for patterns of the following kind:

   NP_h (funcword)+ NP_2(NP_n,)* and|or NP_1

   where NP_h is a baseNP, where *(funcword)+* is one or more function words [5] and where the sequence *'(NP_n, (and|or))+ NP_1'* is a conjoined noun phrase.

[4] This is obviously not optimal; better solutions would be to find the 'best' set (for example, one that composes classes with most members) or to implement all sets.

[5] A function word is here negatively defined as anything but verbs (including auxiliary verbs), adjectives, nouns or full stops.

3. Extract related hypernyms and hyponyms where:

   a. hypernym from step 1. is head in NP_h
   b. hyponym from step 1. is head in one of the noun phrases in the conjoined noun phrase.

7 Results & Evaluation

The number of extracted hh-constructions from the original corpus is 14,828. Statistics describing changes in the data throughout the implementation is presented in Table 3. The table shows the change in number of top nodes as well as in the number of *d-pairs*. A *d-pair* is defined as an ordered pair of terms $< t1, t2 >$ in the hierarchy where t1 dominates t2, and where t1 $\neq$ t2. For example, from the hierarchy in Figure 1 we get eight d-pairs.

The statistics in the last column in Table 3 presents the number of d-pairs per top node in the data. This is suggested as a measurement of how complex the partial hierarchies are on average.

The three values in Table 3 – number of top nodes, number of d-pairs and d-pairs per top node – are given for the original data, the original with the extended data, the classified data and for the data when hierarchical structure is introduced. Values are also given for two possible extracted lexicons (see below).

As can be seen in Table 3, the number of d-pairs increased through the introduction of hierarchies by 2,071 d-pairs (from 22,832 to 24,903 pairs). The relatively low number of new hyponymy relations (that is d-pairs) is disappointing, but with improvements discussed later, the number could hopefully be increased.

Evaluation of semantic hierarchies or lexicon always presents a challenge. Usually, human judges are used to evaluate the result, or the result is compared against a gold-standard resource. Lacking a suitable Swedish gold-standard, our method is evaluated with human judges.

In building a usable lexicon from the data, we try to exclude hierarchies with few terms in them. Several options were tested and two of them are presented in Table 3: one lexicon where all top nodes had at least seven descendants (lexicon-7) and one where all top nodes had at least ten descendants (lexicon-10).

Data	No. of top nodes	No. of d-pairs	d-pairs per top node
Original data	14,828*	24,866	1,68
Orig. + ext. data	15,669*	28,133	1,79
Classification	11,914	22,832	1,92
Hierarchy	11,202	24,903	2,22
lexicon-7	259	5,557	21,45
lexicon-10	154	4,618	29,98

Table 3: Statistics over number of top nodes and d-pairs through the data. * No. of top nodes is equal to the number of hh-constructions.

Human Judge	Percentage of d-pairs judged as correct
judge 1	67.4%
judge 2	52.2%
judge 3	54.1%
judge 4	76.6%

Table 4: Percentage of d-pairs from the partial hierarchies judged as correct by each judge. Total no. of judged d-pairs is 1,000.

The latter, lexicon-10, was used in evaluation. That is, 1,000 of the d-pairs from lexicon-10 was randomly picked in order to evaluate the partial hierarchies and new hyponymy relations. Four human judges were to decide, for each pair, if they thought it was a correct pair or not. The result, presented in Table 4, is in the range of 52.2% to 76.6% correct.

Table 5 presents five ways to look at the result. The first gives the average result over the four judges. The second, *at-least-one*, gives the percentage of d-pairs where at least one of the judges deemed the pair as correct. The *majority* is the percentage of d-pairs where at least two deemed the pair as correct, and the *consensus* option refers to the percentage of d-pairs where all judges agreed. The *at-least-one* option, the least strict of the measures, give us 82.2% correct, while the most strict (the *consensus*) gives us 41.6% correct.

The kappa value (Carletta, 1996) was used to evaluate the agreement among the judges and to estimate how difficult the evaluation task was. Not

average	62.5%
at-least-one	82.2%
majority	71.0%
consensus	41.6%
kappa	K = 0.51

Table 5: Statistics on results from evaluation of 1,000 d-pairs, by four judges.

surprisingly, as evaluation of semantic information, in general, is hard to perform on purely objective grounds, the kappa value is rather low; that is, the value for four annotators on the 1,000 d-pairs is K=0.51. The low kappa value for the evaluation task reflects the great many problems of evaluations of semantic resources by humans. Some of these problems are discussed below:

While lemmatization or stemming is necessary for performing this kind of task, it may also cause problems in cases where morphology is important for correct classification. For example, while the plural form of the word 'boy' (i.e. 'boys') is a valid hyponym of the hypernym 'group', the singular form would not be.

As was also reported by Caraballo (1999), the judges sometimes found proper nouns (as hyponyms) hard to evaluate. E.g. it might be hard to tell if 'Simon Le Bon' is a valid hyponym to the hypernym 'rock star' if his identity is unknown to the judge. One way to overcome this problem might be to give judges information about a sequence of higher ancestors, in order to make the judgement easier.

It is difficult to compare these results with results from other studies such as that of Caraballo (1999), as the data used is not the same. However, it seems that our figures are in the same range as those reported in previous studies.

Charniak & Roark (1998), evaluating the semantic lexicon against gold standard resources (the MUC-4 and the WSJ corpus), reports that the ratio of valid to total entries for their system lies between 20% and 40%.

Caraballo (1999) let three judges evaluate ten internal nodes in the hyponymy hierarchy, that had at least twenty descendants. Cases where judges had problems with proper nouns as hyponyms, corresponding to these mentioned above, were corrected. When the best hypernym was evaluated, the result reported for a majority of the judges was 33%.

8 Discussion and future work

In this paper, we have mainly been concentrating on algorithm development for building the partial hierarchies and on evaluating the quality of the hyponymy relations in the hierarchies. In future work we will continue to put our efforts to include more of the extracted data into the hierarchies.

In classification of hh-construction data (section 6.1), for example, there is a great many classes that are never collapsed where there should have been a collapse. That is, correct sense distinction is captured (through correct collapses), but incorrect sense distinction is also introduced due to lack of overlap in hyponyms. For example, if two classes with the hypernym 'animal' are found where there is no non-empty intersection in hyponyms, 'animal' will incorrectly be treated as having two senses. This is a side effect of the method we are using in order to get disambiguated data to build hierarchies from.

In most cases, introduction of incorrect sense distinction is due to one of two situations: first, when the hypernym only has proper noun hyponyms (e.g. 'person' or 'artist'), the overlap in hyponyms tends to be small. Secondly, when the hypernym is a very general concept, for example 'part', 'question' or 'alternative', the hyponyms will rarely overlap. No assessment of the scope of these problems has been performed in this study. A more thorough investigation ought to be performed in order to know how to overcome the problem of incorrect sense distinctions.

Also, the kind of general, underspecified hypernyms, such as 'question' mentioned above are rarely meaningful as concepts on their own. As discussed by Hearst (1992), more information is needed to solve the underspecification, and the missing information is probably found in previous sentences. An improved algorithm has to deal with this problem – either in excluding this type of hypernyms, or in improving on the concepts by finding information that solves the underspecification.

Modification in the algorithm to impose hierarchi-

cal structure should be carried out in the future, so that more compositions are performed for each class (as discussed in section 6.2). This, together with a more elaborate extension algorithm (section 6.3) should give us further hierarchical links in the lexicon.

Compound analysis and improvements on term extraction for Swedish will also be helpful in future work. Improvements would possibly lead to more collapses by the algorithm presented in section 6.1, which in turn would reduce the number of incorrect sense distinctions.

The resulting hierarchies are not fully strict, e.g. descendants of the same lemma type can occasionally be found in different branches of the same tree. This has to be dealt with in future implementations, as well.

9 Conclusions

We have shown how an unambiguous hypernym-hyponym lexicon with partial hierarchies can be built from data that is unrestricted by domain. The algorithm has been implemented for Swedish, but we can assume that the method easily can be applied to other languages as well. Even though the number of hierarchical structures imposed by the method is rather low, the quality of the hyponymy relations is good and we believe that improvements in the algorithm will increase the number of partial hierarchies.

Acknowledgments

Many thanks to Kjell Elenius for making the corpus of news texts available to me, to the human judges for their patience with the difficult evaluation task, to Rolf Carlson, Beáta Megyesi and Sofia Gustafson-Capková for helpful comments and discussions. Many thanks also for helpful suggestions for improvements from the anonymous reviewers.

This research was supported by CTT's industrial partners, KHT, VINNOVA and the Swedish National Graduate School of Language Technology.

References

Thorsten Brants. 2000. A statistical part-of-speech tagger. *In Proceedings of the 6th Applied Natural Language Processing Conference*, pages 206–213.

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. *In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

Jean Carletta. 1996. Assessing agreement on classification tasks. *Computational Linguistics*, 22(2):249–254.

Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, and Markus Åström. 1992. The linguistic annotation system of the Stockholm-Umeå corpus project. Technical Report no. 33, Dept. of General Linguistics, University of Umeå.

Christiane Fellbaum. 1998. Wordnet: An electronic lexical database. *The MIT Press*.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. *In Proceeding of the Fourteenth International Conference on Computational Linguistics*.

Beáta Megyesi. 2001. Comparing data-driven learning algorithms for PoS tagging of Swedish. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing 2001*, pages 151–158.

Emmanuel Morin and Christian Jacquemin. 1999. Projecting corpus-based semantic links on a thesaurus. *In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. *In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–132.

Brian Roark and Eugene Charniak. 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. *In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116.

Mark Sanderson and W. Bruce Croft. 1999. Deriving structure from text. *In Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–213.

W. A. Woods. 1997. Conceptual indexing: A better way to organize knowledge. Technical Report TR-97-61, Sun Labs.