Learning Semantic Network Patterns for Hypernymy Extraction

Tim vor der Brück
Intelligent Information and Communication Systems (IICS)
FernUniversität in Hagen
tim.vorderbrueck@fernuni-hagen.de

Abstract

Current approaches of hypernymy acquisition are mostly based on syntactic or surface representations and extract hypernymy relations between surface word forms and not word readings. In this paper we present a purely semantic approach for hypernymy extraction based on semantic networks (SNs). This approach employs a set of patterns
$\text{SUB0}(a1, a2) \leftarrow$ *premise* where the premise part of a pattern is given by a SN. Furthermore this paper describes how the patterns can be derived by relational statistical learning following the Minimum Description Length principle (MDL). The evaluation demonstrates the usefulness of the learned patterns and also of the entire hypernymy extraction system.

1 Introduction

A concept is a hypernym of another concept if the first concept denotes a superset of the second. For instance, the class of *animals* is a superset of the class of *dogs*. Thus, animal is a hypernym of its hyponym dog and a hypernymy relation holds between animal and dog. A large collection of hypernymy (supertype) relations is needed for a multitude of tasks in natural language processing. Hypernyms are required for deriving inferences in question answering systems, they can be employed to identify similar words for information retrieval or they can be useful to avoid word-repetition in natural language generation systems. To build a taxonomy manually requires a large amount of work. Thus, automatic approaches for their construction are preferable. In this work we introduce a semantically oriented approach where the hypernyms are extracted using a set of patterns which are neither syntactic nor surface-oriented but instead purely semantic and are based on a SN formalism. The patterns are applied on a set of SNs which are automatically derived from the German Wikipedia[1] by a deep syntactico-semantic analysis. Furthermore, these patterns are automatically created by a machine learning approach based on the MDL principle.

2 Related Work

Patterns for hypernymy extraction were first introduced by Hearst (Hearst, 1992), the so-called Hearst patterns. An example of such a pattern is:
NP_{hypo} {,NP_{hypo}}*{,} and other NP_{hyper}.

These patterns are applied on arbitrary texts and the instantiated variables NP_{hypo} and NP_{hyper} are then extracted as a concrete hypernymy relation.

Apart from the handcrafted patterns there was also some work to determine patterns automatically from texts (Snow and others, 2005). For that, Snow et al. collected sentences in a given text corpus with known hypernym noun pairs. These sentences are then parsed by a dependency parser. Afterwards, the path in the dependency tree is extracted which connects the corresponding nouns with each other. To account for certain key words indicating a hypernymy relation like *such* (see first Hearst pattern) they added the links to the word on either side of the two nouns (if not yet contained) to the path too. Frequently oc-

[1] Note that for better readability the examples are translated from German into English throughout this paper.

Proceedings of the 6th Workshop on Ontologies and Lexical Resources (Ontolex 2010), pages 38–47,
Beijing, August 2010

curring paths are then learned as patterns for indicating a hypernymy relation.

An alternative approach for learning patterns which is based on a surface instead of a syntactic representation was proposed by Morin et al. (Morin and Jaquemin, 2004). They investigate sentences containing pairs of known hypernyms and hyponyms as well. All these sentences are converted into so-called "lexico-syntactic expressions" where all NPs and lists of NPs are replaced by special symbols, e.g.: *NP find in NP such as LIST*. A similarity measure between two such expressions is defined as the sum of the maximal length of common substrings for the maximum text windows before, between and after the hyponym/hypernym pair. All sentences are then clustered according to this similarity measure. The representative pattern (called *candidate pattern*) of each cluster is defined to be the expression with the lowest mean square error (deviation) to all other expressions in the same similarity cluster. The patterns to be used for hyponymy detection are the candidate patterns of all clusters found.

3 MultiNet

MultiNet is an SN formalism (Helbig, 2006). In contrast to SNs like WordNet (Fellbaum, 1998) or GermaNet (Hamp and Feldweg, 1997), which contain lexical relations between synsets, MultiNet is designed to comprehensively represent the semantics of natural language expressions. An SN in the MultiNet formalism is given as a set of vertices and arcs where the vertices represent the concepts (word readings) and the arcs the relations (or functions) between the concepts. A vertex can be lexicalized if it is directly associated to a lexical entry or non-lexicalized. An example SN is shown in Fig. 1. Note that each vertex of the SN is assigned both a unique ID (e.g., *c2*) and a label which is the associated lexical entry for lexicalized vertices and *anon* for non-lexicalized vertices. Thus, two SNs differing only by the IDs of the non-lexicalized vertices are considered equivalent. Important MultiNet relations/functions are (Helbig, 2006):

- AGT: Conceptual role: Agent
- ATTR: Specification of an attribute
- VAL: Relation between a specific attribute and its value
- PROP: Relation between object and property
- *ITMS: Function enumerating a set
- PRED: Predicative concept characterizing a plurality
- OBJ: Neutral object
- SUB0: Relation of conceptual subordination (hyponymy) and hyperrelation to SUBR, SUBS, and SUB
- SUBS: Relation of conceptual subordination (for situations)
- SUBR: Relation of conceptual subordination (for relations)
- SUB: Relation of conceptual subordination other than SUBS and SUBR

MultiNet is supported by a semantic lexicon (Hartrumpf and others, 2003) which defines, in addition to traditional grammatical entries like gender and number, semantic information consisting of one or more ontological sorts and several semantic features for each lexicon entry. The ontological sorts (more than 40) form a taxonomy. In contrast to other taxonomies, ontological sorts are not necessarily lexicalized, i.e., they need not denote lexical entries. The following list shows a small selection of ontological sorts which are inherited from *object*:

- Concrete objects: e.g., *milk*, *honey*
  - Discrete objects: e.g., *chair*
  - Substances: e.g.,, *milk*, *honey*
- Abstract objects: e.g., *race*, *robbery*

Semantic features denote certain semantic properties for objects. Such a property can either be present, not present or underspecified. A selection of several semantic features is given below:

ANIMAL, ANIMATE, ARTIF (artificial), HUMAN, SPATIAL, THCONC (theoretical concept)

Example for the concept *bottle.1.1*[2]: discrete object; ANIMAL -, ANIMATE -, ARTIF +, HUMAN -, SPATIAL +, THCONC -, ...

[2]the suffix *.1.1* denotes the reading numbered *.1.1* of the word *bottle*.

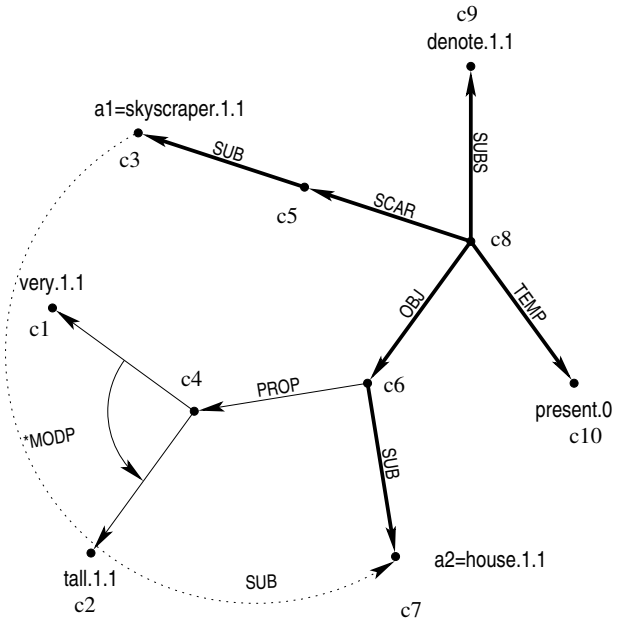

Figure 1: Matching a pattern to an SN. Bold lines indicate matched arcs, the dashed line the inferred arc.

The SNs as described here are automatically constructed from (German) texts by the deep linguistic parser WOCADI[3](Hartrumpf, 2002) whose parsing process is based on a word class functional analysis.

4 Application of Deep Patterns

The extraction of hyponyms as described here is based on a set of patterns. Each pattern consists of a conclusion part SUB0($a1$, $a2$) and a premise part in form of an SN where both *a1* and *a2* have to show up. The patterns are applied by a pattern matcher (or automated theorem prover if axioms are used) which matches the premise with an SN. The variable bindings for *a1* and *a2* are given by the matched concepts of the SN. An example pattern which matches to the sentence: *A skyscraper denotes a very tall building.* is D_4 (see Table 1). The pattern matching process is illustrated in Fig.1. The resulting instantiated conclusion which is stored in the knowledge base is SUB0($skyscraper.1.1, house.1.1$). Advantages by using the MultiNet SN formalism for hypernym (and instance-of relation) acquisition consists of: learning relations between word readings instead of words, the possibility to apply logical axioms and background knowledge, and that person names are already parsed.

An example sentence from the Wikipedia corpus where a hypernymy relation was successfully extracted by our deep approach and which illustrates the usefulness of this approach is: *In any case, not all incidents from the Bermuda Triangle or from other world areas are fully explained.* From this sentence, a hypernymy pair cannot be extracted by the Hearst pattern *X or other Y*. The application of this pattern fails due to the word *from* which cannot be matched. To extract this relation by means of shallow patterns an additional pattern would have to be introduced. This could also be the case if syntactic patterns were used instead since the coordination of *Bermuda Triangle* and *world areas* is not represented in the syntactic constituency tree but only on a semantic level[4].

[3]WOCADI is the abbreviation for **wo**rd **cla**ss **di**sambiguation.

[4]Note that some dependency parsers normalize some syntactic variations too.

5 Graph Substructure Learning By Following the Minimum Description Length Principle

In this section, we describe how the patterns can be learned by a supervised machine learning approach following the Minimum Description Length principle. This principle states that the best hypothesis for a given data set is that one which minimizes the description of the data (Rissanen, 1989), i.e., compresses the data the most. Basically we follow the substructure learning approach of Cook and Holder (Cook and Holder, 1994).

According to this approach, the description length to minimize is the number of bits required to encode a certain graph which is compressed by means of a substructure. If a lot of graph vertices can be matched with the substructure vertices, this description length will be quite small. For our learning scenario we investigate collection of SNs containing a known hypernymy relationship. A pattern (given by a substructure in the premise) which compresses this set quite well is expected to be useful for extracting hypernyms.

Let us first determine the number of bits to encode the entire graph or SN. A graph can be represented by its adjacency matrix and a set of vertex and arc labels. Since an adjacency matrix consists only of ones and zeros, it is well suitable for a binary encoding. For the encoding process, we do not regard the label names directly but instead their number assuming an ordering exists on the label names (e.g., alphabetical).

	$c1$	$c2$	$c3$	$c4$	$c5$	$c6$	$c7$	$c8$	$c9$	$c10$
$c1$	0	0	0	0	0	0	0	0	0	0
$c2$	0	0	0	0	0	0	0	0	0	0
$c3$	0	0	0	0	0	0	0	0	0	0
$c4$	1	1	0	0	0	0	0	0	0	0
$c5$	0	0	1	0	0	0	0	0	0	0
$c6$	0	0	0	0	0	0	1	0	0	0
$c7$	0	0	0	0	0	0	0	0	0	0
$c8$	0	0	0	0	1	1	0	0	1	1
$c9$	0	0	0	0	0	0	0	0	0	0
$c10$	0	0	0	0	0	0	0	0	0	0

Figure 2: Adjacency matrix of the SN.

To encode all labels the number of labels and a list of all label numbers have to be specified, e.g., *3,1,2,1* for 3 vertices with two different label numbers[5] (1,2). The first number encoding (3) starts at position 0 in the bit string, the second (1) at position $2 = \lceil \log_2 3 \rceil$, the third one at position $2 + \lceil \log_2 2 \rceil$, etc. Since the graph actually need not to be encoded in this way but only the length of the encoding is important, non-integer numbers of bits are accepted for simplicity too. If there are a total of l_u different labels, then each encoded label number requires $\log_2(l_u)$ bits. The total number of bits to encode the vertex labels are then given by:
$vbits = \log_2(v) + v \log_2(l_u)$ in which v denotes the total number of vertices[6].

In the next step, the adjacency matrix is encoded where each row is processed separately. A straightforward approach for encoding one row would be to use v number of bits, one for every column. However, the number of zeros are generally much larger than the number of ones which means that a better compression of the data is possible by exploiting this fact. Consider the case that a certain matrix row contains exactly m ones. There are $\begin{pmatrix} v \\ m \end{pmatrix}$ possibilities to distribute the ones to the individual cells. All possible permutations could be specified in a list. In this case it is only necessary to specify the position in this list to uniquely describe one row. Let $b = \max_i k_i$. Then the number of ones in one row can be encoded using $\log_2(b+1)$ bits. $\log_2 \begin{pmatrix} v \\ k_i \end{pmatrix}$ bits are required to encode the distribution of ones in one row. Additionally, $\log_2(b+1)$ bits are needed to encode b which is only necessary once for the matrix. Let us consider the adjacency matrix given in Fig. 2 of the SN shown in Fig. 1 with 10 rows and columns where each row contains at most four ones. To encode the row c_4, containing two ones, re-

[5] The commas are only included for better readability and are actually not encoded.

[6] The approach of Cook and Holder is a bit inexact here. To be precise, the number of bits needed to encode v and b would have to be known a priori.

quires $\log_2(4) + \log_2 \binom{10}{2} = 7.49$ bits which is smaller than 10 bits which were necessary for the naïve approach. The total length *rbits* of the encoding is given by:

$$\begin{aligned} rbits =& \log_2(b+1) + \sum_{i=1}^{v} [\log_2(b+1) + \\ & \log_2 \binom{v}{k_i}] \\ =& (v+1)\log_2(b+1) + \\ & \sum_{i=1}^{v} \log_2 \binom{v}{k_i} \end{aligned} \tag{1}$$

Finally, the arcs need to be encoded. Let $e(i,j)$ be the number of arcs between vertex i and j in the graph and $m := max_{i,j} e(i,j)$. $\log_2(m)$ bits are required to encode the number of arcs between both vertices and $\log_2(l_e)$ bits are needed for the arc label (out of a set of l_e elements). Then the entire number of bits is given by (e is the number of arcs in the graph):

$$\begin{aligned} ebits =& \log_2(m) + \sum_{i=1}^{v}\sum_{j=1}^{v} [A[i,j] log_2(m) + \\ & e(i,j)\log_2(l_e)] \\ =& \log_2(m) + e\log_2(l_e) + \\ & \sum_{i=1}^{v}\sum_{j=1}^{v} A[i,j]\log_2(m) \\ =& e(\log_2(l_e)) + (K+1)\log_2(m) \end{aligned} \tag{2}$$

where K is the number of ones in the adjacency matrix.

The total description length of the graph is then given by: $vbits + rbits + ebits$.

Now let us investigate how the description length of the compressed graph is determined. In the original algorithm the substructure is replaced in the graph by a single vertex. The description length of the graph compressed by the substructure is then given by the description length of the substructure added by the description length of the modified graph.

	$c1$	$c2$	$c3$	$c4$	$c5$	$c6$	$c7$	$c8$	$c9$	$c10$
$c1$	0	0	0	0	0	0	0	0	0	0
$c2$	0	0	0	0	0	0	0	0	0	0
$c3$	0	0	0	0	0	0	0	0	0	0
$c4$	1	1	0	0	0	0	0	0	0	0
$c5$	0	0	×	0	0	0	0	0	0	0
$c6$	0	0	0	0	0	0	×	0	0	0
$c7$	0	0	0	0	0	0	0	0	0	0
$c8$	0	0	0	0	×	×	0	0	×	×
$c9$	0	0	0	0	0	0	0	0	0	0
$c10$	0	0	0	0	0	0	0	0	0	0

Figure 3: Adjacency matrix of the compressed SN. Vertices whose connections can be completely inferred from the pattern are removed.

In our method there are two major differences from the graph learning approach of Cook and Holder.

- Not a single graph is compressed but a set of graphs.
- For the approach of Cook and Holder, it is unknown which vertex of the substructure a graph node is actually connected with. Thus, the description is not complete and the original graph could not be reconstructed using the substructure and the compressed graph. To make the description complete we specify the bindings of the substructure vertices to the graph vertices.

The generalization of the Cook and Holder-algorithm to a set of graphs is quite straight forward. The total description length of a set of compressed graphs is given by the description length of the substructure (here pattern) added to the sum of the description lengths of each SN compressed by this pattern.

Additional bits are needed to encode the vertex bindings (assuming the pattern premise is contained in the SN). First the number of bindings *bin* ($[1, v_p]$, v_p: number of non-lexicalized vertices appearing in a pattern) has to be specified which requires $\log_2(v_p)$ bits. The number of bits needed to encode a single binding is given by $\log_2(v_p) + \log_2(v)$ (vertex indices: $[0, v_p - 1]$ to $[0, v-1]$). Thus, the total

number of required bits is given by

$$binbits = bin(\log_2(v_p) + \log_2(v)) + \log_2(v_p) \quad (3)$$

Note that not all bindings need to be encoded. The number of required binding encodings can be determined as follows. First all bindings for all non-lexicalized pattern vertices are determined. Then all cells from the adjacency matrix of the SN which contain a one and are also contained in the adjacency matrix of the pattern, if this binding is applied to the non-lexicalized pattern vertices, are set to zero. Vertices which contain only zeros in the adjacency matrix on both columns and rows are removed from the adjacency matrix/graph. The arcs from and to this vertex can be completely inferred by the pattern which means that all vertices this vertex is connected with are also contained in the pattern. Since SNs differing only by the IDs of their non-lexicalized vertices are considered identical, no binding has to be specified for such a vertex. Additionally, the modified adjacency matrix is the result of the compression by the pattern, i.e., vbits, rbits, and ebits are determined from the modified adjacency matrix/graph if the pattern was successfully matched to the SN.

Let us consider our example pattern D_4 (Table 1). The following bindings are determined: a1: c3 (a1); a: c8; c: c6; b: c5; a2: c7 (a2)

The bindings for *a1* and *a2* need not to be remembered since all hyponym vertices are renamed to *a1* and the hypernym vertices to *a2* in order to learn generic patterns for arbitrary hypernyms/hyponyms. The cells of the adjacency matrix which are associated to the arcs: $\text{SCAR}(c8, c5)$, $\text{SUB}(c5, a1)$, $\text{OBJ}(c8, c6)$, $\text{SUBS}(c8, c9)$, $\text{TEMP}(c8, c10)$ are set to zero (marked by a cross in Fig. 3) since these arcs are also represented in the pattern using the bindings stated above. The rows and columns of $c3$, $c5$, $c7$, and $c9$ of the modified graph adjacency matrix only contain zeros. Thus, these rows can be removed from the adjacency matrix and the associated concepts can be eliminated from the vertex set of the SN.

The findings of the optimal patterns is done compositionally employing a beam search approach. First this approach starts with patterns containing only a single arc. These patterns are then extended by adding one arc after another preferring patterns leading to small description lengths of the compressed SNs. Note that only pattern premises are allowed which are fully connected, e.g., $\text{SUB}(a, c) \wedge \text{SUB}(e, f)$ is no acceptable premise.

Two lists are used during the search, $local\_best_i$ for guiding the search process and *global_best* for storing the best global results found so far:

- $local\_best_i$: The k best patterns of length i
- *global_best*: The k best patterns of any length

The list $local\_best_i$ is determined by extending all elements from $local\_best_{i-1}$ by one arc and only keeping the k arcs leading to the smallest description length. The list *global_best* is updated after each change of the list $local\_best_i$. This process is iterated as long as the total description length can be further reduced, i.e., $DL(local\_best_{i+1}[0]) < DL(local\_best_i[0])$, where $DL : Pattern \rightarrow \mathbb{R}$ denotes the description length of a pattern and [0] accesses the first element of a list.

The list *global_best* contains as the result of this approach the k patterns with the smallest overall compressed description length[7]. Note however that it is often not recommended to use all elements of *global_best* since this list contains oftentimes patterns where the premise part is a subgraph (can be inferred by) another premise pattern part contained in this list and their combination would actually not reduce the description length. Thus, in addition to the original approach of Cook and Holder, a dependency resolution is done.

The following iterative approach is proposed to cancel out such dependent patterns:

1. Start with the first entry of the global list: $depend\_best := \{global\_best[0]\}$

[7] compressed description length: short for description length of the SNs compressed by the pattern

ID	Definition	Matching Expression
D_1	$\text{SUB}0(a1, a2) \leftarrow$ $\text{SUB}(g, a2) \wedge \text{ATTCH}(g, f) \wedge$ $\text{SUBR}(e, sub.0) \wedge \text{TEMP}(e, present.0) \wedge$ $\text{ARG}2(e, f) \wedge \text{ARG}1(e, d) \wedge$ $\text{SUB}(d, a1)$	An apple$_{hypo}$ is a type of fruit$_{hyper}$.
D_2	$\text{SUB}0(a1, a2) \leftarrow$ $\text{SUB}(f, a2) \wedge \text{EQU}(g, f) \wedge$ $\text{SUBR}(e, equ.0) \wedge \text{TEMP}(e, present.0) \wedge$ $\text{ARG}2(e, f) \wedge \text{ARG}1(e, d) \wedge$ $\text{SUB}(d, a1)$	Psycho-linguistics$_{hypo}$ is a science$_{hyper}$ of the human ability to speak.
D_3	$\text{SUB}0(a1, a2) \leftarrow$ $\text{PRED}(g, a2) \wedge \text{ATTCH}(g, f) \wedge$ $\text{SUBR}(e, pred.0) \wedge \text{ARG}2(e, f) \wedge$ $\text{TEMP}(e, present.0) \wedge \text{ARG}1(e, d) \wedge$ $\text{PRED}(d, a1)$	Hepialidae$_{hypo}$ are a kind of insects$_{hyper}$. literal translation from: *Die Wurzelbohrer sind eine Familie der Schmetterlinge.*
D_4	$\text{SUB}0(a1, a2) \leftarrow$ $\text{SUB}(f, a2) \wedge \text{SUBS}(e, denote.1.1) \wedge$ $\text{TEMP}(e, present.0) \wedge \text{OBJ}(e, f) \wedge$ $\text{SCAR}(e, d) \wedge \text{SUB}(d, a1)$	A skyscraper$_{hypo}$ denotes a very tall building$_{hyper}$.
D_5	$\text{SUB}0(a1, a2) \leftarrow$ $\text{PROP}(f, other.1.1) \wedge \text{PRED}(f, a2) \wedge$ $foll_{*_{\text{ITMS}}}(d, f) \wedge \text{PRED}(d, a1)$	ducks$_{hypo}$ and other animals$_{hyper}$
D_6	$\text{SUB}0(a1, a2) \leftarrow$ $\text{SUB}(d, a2) \wedge \text{SUB}(d, a1)$	the instrument$_{hyper}$ cello$_{hypo}$
D_7	$\text{SUB}0(a1, a2) \leftarrow \text{SUB}(f, a2) \wedge$ $\text{TEMP}(e, present.0) \wedge \text{SUBR}(e, sub.0) \wedge$ $\text{SUB}(d, a1) \wedge \text{ARG}2(e, f) \wedge$ $\text{ARG}1(e, d)$	The Morton number$_{hypo}$ is a dimensionless indicator$_{hyper}$.

Table 1: A selection of automatically learned patterns.

2. Set *index*:=1
3. Calculate the combined (compressed) description length of *depend_best* and $\{global\_best[index]\}$
4. If the combined description length is reduced add $global\_best[index]$ to *depend_best*, otherwise leave *depend_best* unchanged
5. If $counter \geq length(global\_best)$ then return *depend_best*
6. $index := index + 1$
7. Go back to step 3

6 System Architecture

In this section, we give an overview over our hypernymy extraction system. The following procedure is employed to identify hypernymy relations in Wikipedia (see Fig. 4):

1. At first, all sentences of Wikipedia are analyzed by the deep analyzer WOCADI (Hartrumpf, 2002). As a result of the parsing process, a token list, a syntactic dependency tree, and an SN is created.

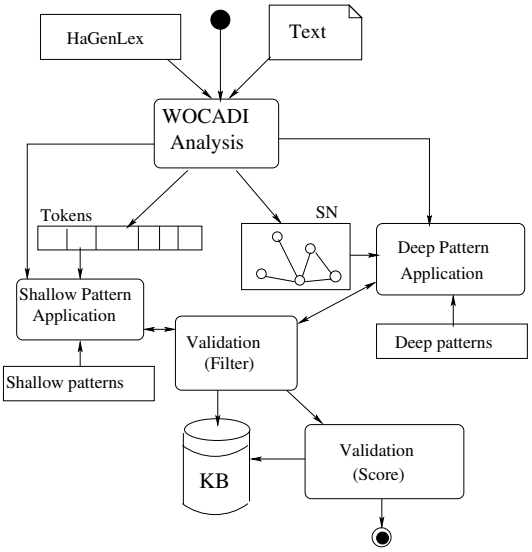

Figure 4: Activity diagram of the hypernym extraction process.

2. Shallow patterns based on regular expressions are applied to the token lists, and deep patterns (learned and hand-crafted) are applied to the SNs to generate proposals for hypernymy relations.
3. A validation tool using ontological sorts and semantic features checks whether the proposals are technically admissible at all to reduce the amount of data to be stored in the knowledge base KB.
4. If the validation is successful, the hypernymy hypothesis is integrated into KB. Steps 2–4 are repeated until all sentences are processed.
5. Each hypernymy hypothesis in KB is assigned a confidence score estimating its reliability.

7 Validation Features

The knowledge acquisition carried out is followed by a two-step validation. In the first step, we check the ontological sorts and semantic features of relational arguments for subsumption. For instance, a discrete concept (ontological sort: d) denoting a human being (semantic feature: human +) can only be hypernym of an other object, if this object is both discrete and a human being as well. Only relational candidates for which semantic features and ontological sorts can be shown to be compatible are stored in the knowledge base.

In a second step, each relational candidate in the knowledge base is assigned a quality score. This is done by means of a support vector machine (SVM) on several features. The SVM determines the classification (hypernymy or non-hypernymy) and a probability value for each hypernymy hypothesis. If the classification is 'hypernymy', the score is defined by this probability value, otherwise as one minus this value.

Correctness Rate: The feature *Correctness Rate* takes into account that the assumed hypernym alone is already a strong indication for the correctness or incorrectness of the investigated relation. The same holds for the assumed hyponym as well. For instance, relation hypotheses with hypernym *liquid* and *town* are usually correct. However, this is not the case for abstract concepts. Moreover, movie names are often extracted incompletely since they can consist of several tokens. Thus, this indicator determines how often a concept pair is classified correctly if a certain concept shows up in the first (hyponym) or second (hypernym) position.

Frequency: The feature *frequency* regards the quotient of the occurrences of the hyponym in other extracted relations in hyponym position and the hypernym in hypernym position.

This feature is based on two assumption. First, we assume that general terms normally occur more frequently in large text corpora than very specific ones (Joho and Sanderson, 2007). Second, we assume that usually a hypernym has more hyponyms than vice-versa.

Context: Generally, the hyponym can appear in the same textual context as its hypernym. The textual context can be described as a set of other concepts (or words for shallow approaches) which occur in the neighborhood of the investigated hyponym/hypernym candidate pair investigated on a large text corpus. Instead of the textual context we regard the semantic context. More specifically, the distributions of all concepts are regarded which are connected with the assumed hypernym/hyponym concept by the MultiNet-PROP (property) relation. The formula to estimate the similarity was basically taken from (Cimiano and others, 2005).

ID	Precision	First Sent.	# Matches
D_1	0.275	0.323	5 484
D_2	0.183	0.230	35 497
D_3	0.514	0.780	937
D_4	0.536	0.706	1 581
D_5	0.592	-	3 461
D_6	0.171	0.167	37 655

Table 2: Precision of hypernymy hypotheses extracted by patterns without usage of the validation component (D_7 not yet evaluated).

See (vor der Brück, 2010) for a more de-

Score	≥0.95	≥0.90	≥0.85	≥0.80	≥0.75	≥0.70	≥0.65	≥0.60	≥0.55
Precision	1.0000	0.8723	0.8649	0.8248	0.8203	0.7049	0.6781	0.5741	0.5703

Table 3: Precision of the extracted hypernymy relations for different confidence score intervals.

tailed description of the validation features.

8 Evaluation

We applied the pattern learning process on a collection of 600 SN, derived by WOCADI from Wikipedia, which contain hyponymically related concepts. Table 1 contains some of the extracted patterns including a typical expression to which this pattern could be matched. The predicate $foll_f(a, b)$ used in this table specifies that argument a precedes argument b in the argument list of function f. Patterns D_1-D_4 and D_7 contain concept definitions where the defined concept is, in many cases, the hyponym of the defining concept. In pattern D_1 and D_7 the defining concept is directly identified by the parser as hypernym of the defined concept (SUBR(e, *sub.0*)). In pattern D_2 the defining concept is recognized as equivalent to the defined concept (SUBR(e, *equ.0*)). However, in most of the cases the defining concept consists of a meaning molecule, i.e., a complex concept where some inner concept is modified by an additional expression (often a property or an additional subclause). If this expression is dropped which is done by the pattern D_2 the remaining concept becomes a hypernym of the defined concept. Pattern D_5 is a well-known Hearst pattern. Pattern D_6 is used to match to appositions. However, for that the representation of appositions in the SN, as provided by the parser, could be improved since the order of the two concepts in a sentence is not clear by regarding only the SN, i.e., from the expression *the instrument cello* both SUB0(*instrument.1.1*, *cello.1.1*) and SUB0(*cello.1.1*, *instrument.1.1*) could be extracted. The incorrect relation hypothesis has to be filtered out (hopefully) by the validation component. A better representation would be by employing the TUPL*($c_1, \ldots, c_n$) predicate which combines several concepts with regard to their order. So the example expression should better be represented by SUB(d, e) ∧ TUPL*(e, *instrument.1.1*, *cello.1.1*).

Precision values for the hyponymy relation hypotheses extracted by the learned patterns, which are applied on a subset of the German Wikipedia, are given in Table 2. The first precision value specifies the overall precision, the second the precision if only hypernymy hypotheses are considered which were extracted from first sentences of Wikipedia articles. The precision is usually increased considerably if only such sentences are regarded. Note that this precision value was not given for pattern D_5 which usually cannot be matched to such sentences. The last number specifies the total amount of sentences a pattern could be matched to.

Furthermore, besides the pattern extraction process, the entire hypernymy acquisition system was validated, too. In total 391 153 different hypernymy hypotheses were extracted employing 22 deep and 19 shallow patterns. 149 900 of the relations were only determined by the deep but not by the shallow patterns which shows that the recall can be considerably increased by using deep patterns in addition. But also precision profits from the usage of deep patterns. The average precision of all relations extracted by both shallow and deep patterns is 0.514 that is considerably higher than the average precision for the relations only extracted by shallow patterns (0.243).

The correctness of an extracted relation hypothesis is given for several confidence score intervals in Table 3. There are 89 944 concept pairs with a score above 0.7, 3 558 of them were annotated with the information of whether the hypernymy relation actually holds.

Note that recall is very difficult to specify since for doing this the number of hypernymy relations which are theoretically extractable

from a text corpus has to be known where different annotators can have very dissenting opinions about this number. Thus, we just gave the number of relation hypotheses exceeding a certain score. However the precision obtained by our system is quite competitive to other approaches for hypernymy extraction like the one of Erik Tjong and Kim Sang which extracts hypernyms in Dutch (Tjong and Sang, 2007) (Precision: 0.48).

9 Conclusion and Outlook

We showed a method to automatically derive patterns for hypernymy extraction in form of SNs by following the MDL principle. A list of such patterns together with precision and number of matches were given to show the usefulness of the applied approach. The patterns were applied on the Wikipedia corpus to extract hypernymy hypotheses. These hypotheses were validated using several features. Depending on the score, an arbitrary high precision can be reached. Currently, we determine confidence values for the precision values of the pattern example. Further future work includes the application of our learning algorithm to larger text corpora in order to find additional patterns. Also an investigation of how this method can be used for other types of semantic relations is of interest.

Acknowledgements

We want to thank all of our department which contributed to this work, especially Sven Hartrumpf and Alexander Pilz-Lansley for proofreading this paper. This work was in part funded by the DFG project *Semantische Duplikatserkennung mithilfe von Textual Entailment* (HE 2847/11-1).

References

Cimiano, P. et al. 2005. Learning taxonomic relations from heterogeneous sources of evidence. In Buitelaar, P. et al., editors, *Ontology Learning from Text: Methods, evaluation and applications*, pages 59–73. IOS Press, Amsterdam, The Netherlands.

Cook, D. and L. Holder. 1994. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255.

Fellbaum, C., editor. 1998. *WordNet An Electronic Lexical Database*. MIT Press, Cambridge, Massachusetts.

Hamp, B. and H. Feldweg. 1997. Germanet - a lexical-semantic net for german. In *Proc. of the ACL workshop of Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, Madrid, Spain.

Hartrumpf, S. et al. 2003. The semantically based computer lexicon HaGenLex – Structure and technological environment. *Traitement automatique des langues*, 44(2):81–105.

Hartrumpf, S. 2002. *Hybrid Disambiguation in Natural Language Analysis*. Ph.D. thesis, Fern-Universität in Hagen, Fachbereich Informatik, Hagen, Germany.

Hearst, M. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING*, Nantes, France.

Helbig, H. 2006. *Knowledge Representation and the Semantics of Natural Language*. Springer, Berlin, Germany.

Joho, H. and M. Sanderson. 2007. Document frequency and term specificity. In *Proc. of RIAO*, Pittsburgh, Pennsylvania.

Morin, E. and C. Jaquemin. 2004. Automatic acquisition and expansion of hypernym links. *Computers and the Humanities*, 38(4):363–396.

Rissanen, J. 1989. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, Hackensack, New Jersey.

Snow, R. et al. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, Cambridge, Massachusetts.

Tjong, E. and K. Sang. 2007. Extracting hypernym pairs from the web. In *Proceedings of the 45 Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Prague, Czech Republic.

vor der Brück, T. 2010. Hypernymy extraction using a semantic network representation. *International Journal of Computational Linguistics and Applications (IJCLA)*, 1(1).