Application of Machine Learning Algorithms for Automatic Knowledge Acquisition and Readability Analysis

Technical Report

Tim vor der Brück

FernUniversität in Hagen
Intelligent Information and Communication Systems (IICS)
Universitätsstraße 1
58084 Hagen
Germany

E-Mail:
tim.vorderbrueck@fernuni-hagen.de

A large knowledge base is a prerequisite for a lot of tasks in natural language processing (NLP). To build a handcrafted knowledge base, which is applicable to real world scenarios, a vast amount of effort is required. Furthermore, experts are needed with a strong background in linguistics, artificial intelligence and knowledge representation which may not be available to the extent necessary (knowledge acquisition bottleneck). For these reasons, machine learning techniques are widely used to construct a knowledge base automatically. Learning techniques are also relevant to many other areas, e.g., for readability analysis. In the latter area, a lot of work is needed to find the optimal settings for a readability formula and it usually involves a large amount of trial and error iterations. Thus, it is preferable to learn the necessary parameter settings automatically. This report investigates the application of machine learning techniques in both areas. Finally, several freely available machine learning tools, which can be employed to accomplish both tasks, are introduced and compared with each other.

Contents

1 Introduction

Machine learning techniques are widely used in the area of NLP. This report concentrates on two application scenarios: knowledge acquisition and readability analysis.

Knowledge Acquisition: A large knowledge base is a prerequisite for the successful application of NLP technology to a wide range of tasks. A knowledge base can contain several types of knowledge:

- Lexical and morphological knowledge (e.g., part of speech information, named entities),
- Syntactic knowledge (e.g., subcategorization frames), and
- Semantic knowledge (e.g., common sense knowledge including semantic relations and entailments between concepts, and valency frames, i.e., subcategorization frames including semantic information).

The knowledge about commonly used *named entities* can support the parsing process and the coreference resolution. Another application scenario where such knowledge is useful is the automatic generation of hyperlinks for webpages. The approach of Busemann et al. highlights all named entities of certain semantic types occurring in a text [BDK$^+$03]. If the user selects one of the highlighted named entities, a database is queried for a relevant webpage. In case of success, this webpage is displayed in an external browser window.

Valency frames specify the number and type of arguments for verbs, nouns, and adjectives as well as their syntactic and semantic properties. Examples for syntactic properties are the grammatical category or the case of noun phrase arguments. Semantic properties are for instance semantic features or ontological sorts [Hel06]. The knowledge about a valency frame is important for linguistic parsers to resolve ambiguities of syntactic structures of analyzed sentences and of word readings (Word Sense Disambiguation) [Har03].

Important *semantic relations* are hypernymy/hyponymy (supertype and subtype), synonymy, antonymy, and meronymy/holonymy (part-whole relations). Semantic relations are vital for all NLP tasks which require inferences like question answering or recognizing textual entailment [RDH$^+$83, BM06]. Moreover, such relations can be relevant in other areas of NLP as well. For example, a text generation component might replace a concept by a synonym or a hypernym in order to avoid word repetitions for a better writing style [KD96]. Semantic relations are further employed for resolving bridging references which is required for the assimilation of semantic networks [GHH06].

An *entailment* is a logical relation between two formulas such that all models for the first formula (A) are also models for the second (B): $A \models B$. Entailments can be used to represent inferences occurring in natural language. Both, semantic relations between generic concepts and entailments belong to the so-called *common sense knowledge*. Question answering systems require a vast amount of common sense knowledge for a good performance.

The knowledge as mentioned above can either be stored directly in the knowledge base or alternatively in a semantic lexicon depending on the design of the employed NLP system. The manual construction of large knowledge bases or semantic lexicons needs a vast amount of effort. Thus, a variety of methods are proposed to automatically extract the above-mentioned knowledge via text mining over very large corpora. This report investigates machine learning algorithms which are suitable and often used for this purpose.

Readability Analysis: To connect theoretical insights into learning techniques with experiences of their applications in special fields, an overview is given of their usage in the area of readability analysis. Such an analysis assigns a numerical readability score to a given text. Normally, this score is determined by combining several linguistic readability indicators by a weighted sum. Each indicator (e.g., sentence length or word length) assesses a certain aspect of readability.

Structure of this paper: Before going into details, a short overview of the general organization of this document is given. Section 2 introduces the most important machine learning algorithms which are usually employed in the area of knowledge acquisition and readability analysis. Section 3 describes several knowledge extraction methods. In Section 4, typical approaches for learning readability scores are discussed. Freely available machine learning tools, suitable for the above mentioned application scenarios, are presented in Section 5. Finally, a conclusion and outlook are given in Section 6.

2 Machine Learning Algorithms Relevant to Knowledge Acquisition or Readability Analysis

2.1 Naïve Bayes Classifier

The naïve Bayes classifier is one of the simplest classifiers for supervised learning. This classifier applies to learning tasks where each element of a class is described by a conjunction of attribute values where the number of classes is finite [Mit97, p.177-178]. Such a classifier is trained by presenting a set of typical examples; on this basis it predicts the class for previously unseen instances. That class should be chosen for such an instance which maximizes the posteriori probability given the attribute value pairs $(A_1 = a_1, \dots, A_n = a_n)$ of the instance by following the maximum likelihood principle:

$$c_{max} := arg \max_c P(C = c|A_1 = a_1, \dots, A_n = a_n) \tag{1}$$

with:

- A_i: attribute i
- a_i: value of attribute i for the given instance
- C: classification
- c: classification value

where A_i $(i = 1, \dots, n)$ and C are random variables. A straightforward approach could be to approximate the probabilities by means of a simple frequency analysis. A problem normally connected to this procedure arises from potential data sparsity, i.e., if the number of attributes is large, there might be no single training example with the given attribute values a_1,...,a_n. Thus, a procedure, which makes special assumptions about the nature of the given data, is proposed to overcome this difficulty.

Applying Bayes Theorem, Equation 1 can be rewritten as follows:

$$c_{max} := arg \max_c \frac{P(A_1 = a_1, \dots, A_n = a_n|C = c)P(C = c)}{P(A_1 = a_1, \dots, A_n = a_n)} \tag{2}$$

For solving this maximization problem, only the class c is actually varied while the attribute values $a_1, \dots, a_n$ remain unchanged. Thus, Equation 2 can be simplified to:

$$c_{max} := arg \max_c P(A_1 = a_1, \ldots, A_n = a_n | C = c) P(C = c) \tag{3}$$

The naïve Bayes classifier makes the (naïve) assumption that the attributes $A_1, \ldots, A_n$ are statistically independent of each other. In this case, $P(A_1 = a_1, \ldots, A_n = a_n | C = c)$ is given as the product of $P(A_i = a_i | C = c)$ for $i = 1, \ldots, n$. Thus:

$$c_{max} := arg \max_c P(C = c) \prod_{i=1}^{n} P(A_i = a_i | C = c) \tag{4}$$

c_{max} can be found by approximating the probabilities by relative frequencies. Data sparsity is not critical anymore since the relative frequencies of *single* attribute values (instead of a combination of several attribute values) are determined.

The naïve Bayes classifier is used by Cimiano et al. to combine several methods for learning hyponymy relations (see page 20). For that, the attributes represent the outcomes of the different methods and the classes represent the fact whether a hyponymy relation holds or not.

2.2 Artificial Neural Networks

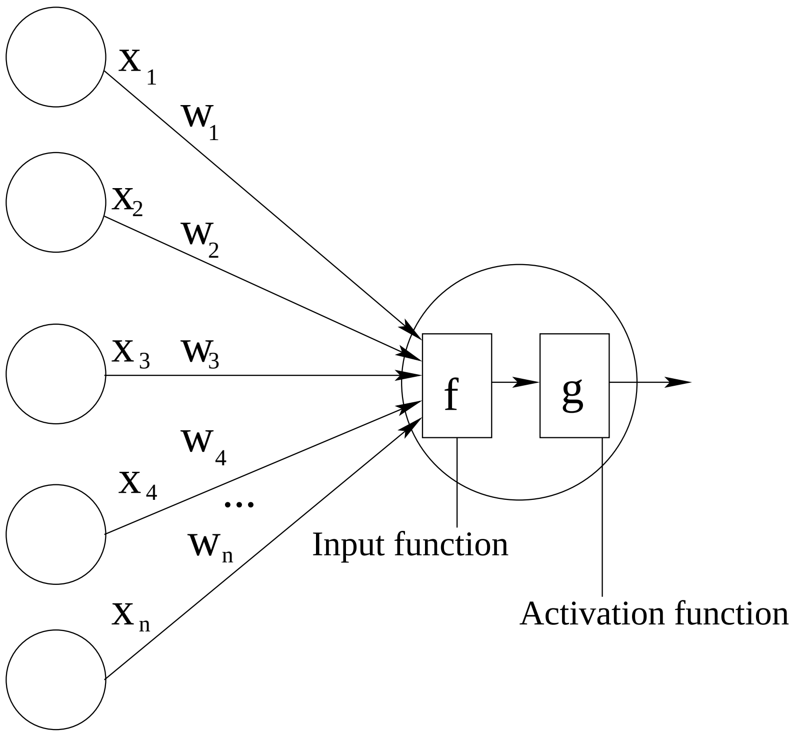


Figure 1: A single perceptron of an ANN.

Artificial neural networks (ANNs) provide a general practical method by learning real-valued, discrete-valued and vector-valued functions from examples and are inspired by the neural networks of a human brain [Mit97]. In this report, we will concentrate on perceptron-based ANNs only (a single perceptron is illustrated in Figure 1). Such an ANN consists of several perceptrons which are connected by directed edges. The edges describe the data flow in the network, i.e., if two perceptrons are connected, the output of the source perceptron is used as the input of the destination perceptron. Additionally, some of the input nodes of a perceptron can have constant values. Furthermore, each edge in the ANN is associated to a real-valued weight. The output of a single perceptron is determined by applying the composition of the activation function and the input function to the input values. The input function is usually given as the weighted sum $f(x) = \sum_{i=1}^{n} x_i w_i$ of all input values x_i. An activation function often used is the *Fermi function*: $g_{\mu,\delta}(f(x)) = \frac{1}{1+e^{-\frac{f(x)-\mu}{\delta}}}$ [Mac05].

A single perceptron is frequently employed for data fusion of several models. This approach is followed by Cimiano et al. [CPSTS05] to combine several methods for hyponymy relation detection (see page 20).

2.3 Decision Tree Algorithm

Decision tree algorithms are supervised machine learning techniques which construct a tree for classifying given data [Mit97] where the data has to be discrete-valued. However, there exist several approaches to automatically derive (discrete) intervals from continuous data which makes it possible to apply this algorithm to continuous data as well. The inner nodes of the tree are associated to attributes, the edges are associated to the values of these attributes and the leaf nodes are labeled with the classes to which the data is assigned to. In order to classify a single data instance the tree is traversed top-down and at each step, the attribute associated to a tree node is compared with the same attribute of the given instance. The edge that matches the attribute value of the given instance is traversed. If a leaf node is reached, the data instance is assigned to the class denoted by the leaf node's label.

In the following, a typical algorithm to learn such a decision tree is described. Note that actually a vast amount of different approaches to accomplish this task exist in practice. A decision tree is usually constructed top-down. At each node it is first checked if all training examples which reach this node in a classification process are classified identically. In this case, the node is marked as a leaf node and labeled with the class of these examples. Otherwise, the attribute most useful for classifying the examples has to be determined. Commonly used scores for determining the best attribute are the *information gain* and the *gain ratio* [Mit97]. Child nodes are created for each value of the chosen attribute. In the best case, examples assigned to different classes are also associated to different values of this attribute. In this case, no further expansion of the child nodes is necessary and all examples will be classified correctly. Otherwise, the expansion of the tree has to be continued, i.e., the process is repeated for all child nodes.

In case no further attribute is left, i.e., all attributes were already consumed by some ancestor node, the current node is also labeled as a leaf node and its class label is determined by a majority vote among the classes of the associated examples.

A decision tree is often used to combine several models with each other (in a quite similar way as Naïve Bayes and ANN, see Sections 2.1 and 2.2). This approach is followed by Cimiano et al. [CPSTS05] for detecting hyponyms (see page 20). Furthermore, the approach of Girju et al. [Fel98] for meronymy extraction is also based on a decision tree classifier (see page 25).

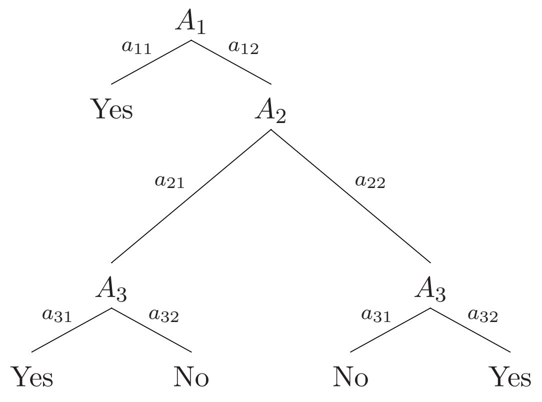


Figure 2: Example of a decision tree with attributes A_1, A_2, A_3. All instances are classified to either *Yes* or *No*.

2.4 Clustering

Clustering is an unsupervised machine learning algorithm. It divides a set of vectors into several (usually disjoint) subsets (called clusters) [MS99, p.495–528]. The elements of the same cluster should be located close to each other in a vector space according to a certain similarity measure (which can be given, for instance, by the Euclidean distance).

Clustering can be divided into hard and soft clustering where for hard clustering a vector always completely belongs to a single cluster. For soft clustering in contrast, the vectors are often assigned a degree of membership to a cluster, i.e., a vector belongs to a cluster with a certain probability.

The clusters can be organized in a flat or hierarchical way where in the latter case, the elements of each cluster are also members of the superior clusters. Hierarchical clusters can be constructed either bottom-up or top-down where the first approach is more widely used. In the area of knowledge acquisition, clustering algorithms are used for extracting hyponymy relations (see Section 3.4) and paraphrases (see Section 3.1).

2.5 Support Vector Machines

A support vector machine (SVM) [Vap79] is an algorithm to divide a set of vectors belonging to two classes by a hyperplane h in such a way that the vectors on the same side of the hyperplane are ideally classified identically (see Figure 3) and the margin of this hyperplane is maximized. The margin is defined as the parallel hyperplanes on either side of h which still separate the data and which have maximum and identical distance to h. The vectors which are located on the margin are called the support vectors. A support vector machine is also able to separate vectors which are not linearly separable by transferring them into a higher dimensional space.

Previously unseen data are distributed into the two classes depending on which side of the hyperplane the corresponding data vector is located. By combining several hyperplanes, a classification into more than two classes is also possible. In the area of knowledge acquisition, a support vector machine is employed for instance for meronymy detection by Aramaki et al [AIMO07] (an description of this approach is given at page 25). For readability analysis, Larsson [Lar06] proposes an approach to

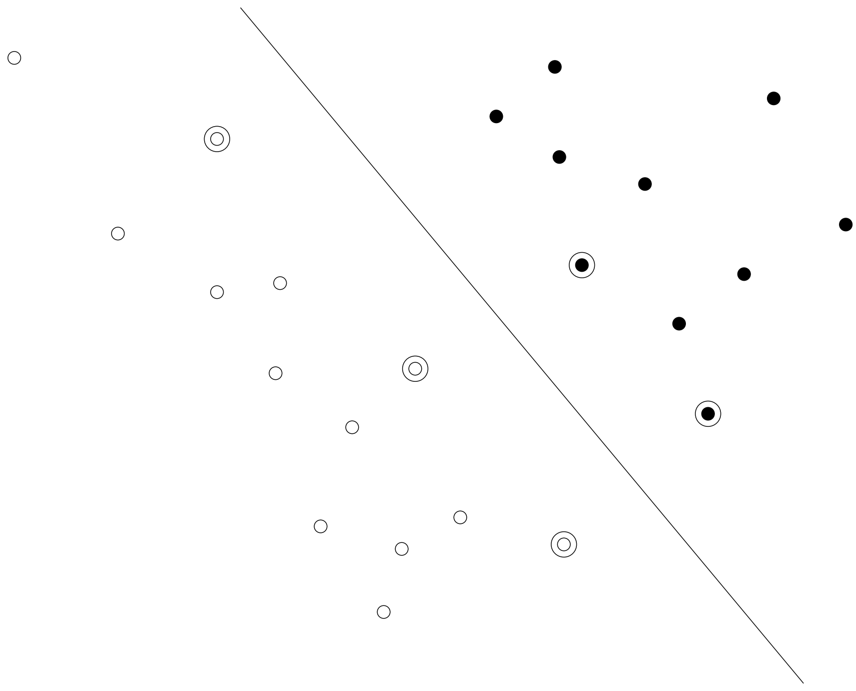


Figure 3: Example for a classification using a support vector machine. A hyperplane divides the set of black points from the set of white points. The support vectors are marked by surrounding circles.

classify texts of a given corpus into several readability levels using SVM technology (see page 31 of this report).

2.6 Genetic Algorithms

Genetic algorithms are a special kind of optimization algorithms which are inspired by biology and are a subclass of evolutionary algorithms [Gol89]. Genetic algorithms are capable of finding a solution for problems which cannot be solved analytically. They start with an initial set of first guesses (suggestions) which are often created by random. The suggestions (also called the population) are continually improved by applying the following operations:

- Selection: successful suggestions (according to a fitness function) are kept, while non-successful suggestions are removed.
- Mutation: create additional suggestions by modifying existing ones.
- Crossover: create additional suggestions by combining existing ones.

An approach of learning graph structures using genetic algorithms is proposed by [Mas94]. The crossover operations employed by this algorithm is illustrated in Figure 4, the mutation operation in Figure 5. A genetic graph learning algorithm could be employed to determine typical MultiNet substructures of sentences containing hypernymy or meronymy relations.

2.7 Linear Regression

Many types of models describe a certain variable (the explained variable) by a linear combination of other variables (the explanatory variables). In the area of readability

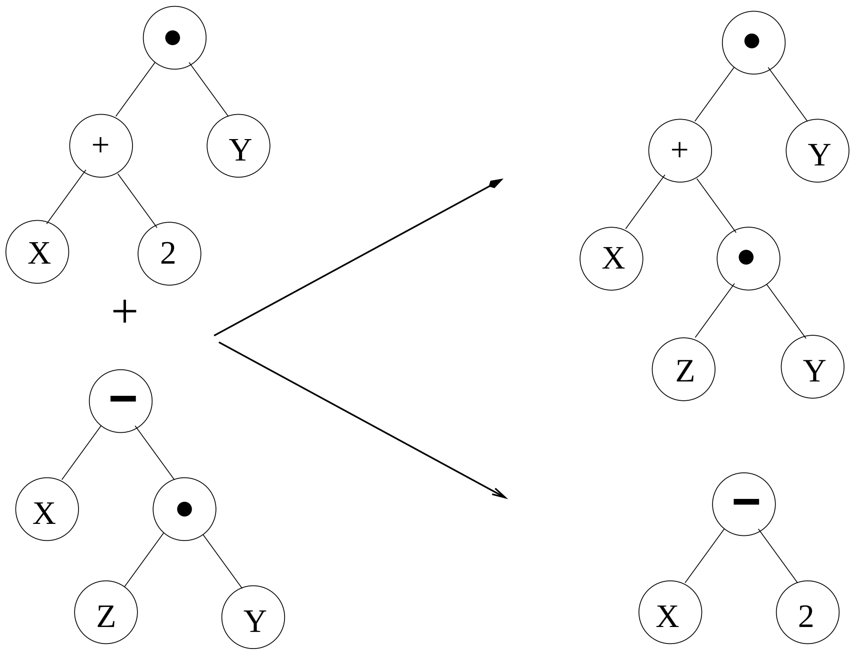


Figure 4: Crossover operations for two graph structures. The resulting graph structures are displayed on the right side.

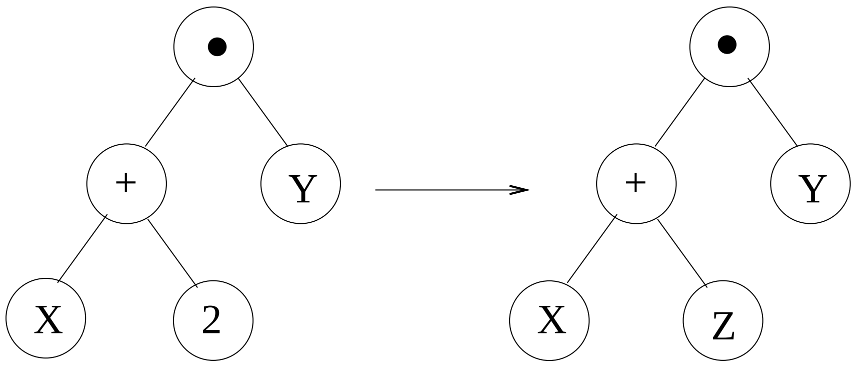


Figure 5: A graph structure is mutated by changing one node symbol.

formulas for instance, a numerical representation of the readability of a text is often calculated by a linear combination of readability indicators like sentence length or word length. Linear regression [Gre93] can be used to determine the parameter vector $\mathbf{w}$ of such a linear combination $\hat{\mathbf{y}} = \mathbf{a} + \mathbf{w}\mathbf{X}$ by minimizing the square error between $\hat{\mathbf{y}}$ and $\mathbf{y}$, where:

- $\mathbf{y}$: vector of the explained variables (desired outcome).
- $\mathbf{X}$: matrix of the explanatory variables. One row of X contains all values belonging to a certain explained variable y_i.
- $\mathbf{w}$: vector of parameters to estimate.
- $\mathbf{a}$: vector of constants.

Additionally, several equality constraints can be defined. In matrix notation they are given as:

$$\mathbf{L}\mathbf{w} = \mathbf{q} \tag{5}$$

where

- $\mathbf{L}$: L_{ij} is the ith coefficient of w_j
- $\mathbf{q}$: result vector

The whole optimization problem can be rewritten as:

$$\mathbf{W} = \begin{bmatrix} \mathbf{X^T X} & \mathbf{L^T} \\ \mathbf{L} & 0 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} \mathbf{X^T y} \\ q \end{bmatrix} \tag{6}$$

$$\mathbf{W} \begin{bmatrix} \mathbf{w} \\ \lambda \end{bmatrix} = \mathbf{u} \tag{7}$$

where λ is the Lagrange Multiplier (it can normally be ignored). Equation 7 can be solved for $\mathbf{w}$:

$$\begin{bmatrix} \mathbf{w} \\ \lambda \end{bmatrix} = \mathbf{W^{-1}u} \tag{8}$$

If no value restrictions are used, Equation 8 simplifies to:

$$\mathbf{w} = (\mathbf{X^T X})^{-1}\mathbf{X^T y} \tag{9}$$

Linear regression is widely used to determine parameters of readability formulas (see Section 4.1).

2.8 Robust Regression

One disadvantage of linear regression is that due to minimizing the square error this method is quite sensitive to outliers [Rya97]. Frequently, an outlier cannot be filtered out by normal outlier detection algorithms since the associated coordinates of this outlier might not differ strongly from the other data points in absolute terms but only regarding to its fit to the target function. Thus, several variants were developed which are more robust to outliers than linear regression and which are therefore named *robust regression*. Robust regression is used to determine the parameters of the DeLite readability formula (see Section 4.1).

2.9 Relational Learning

Relational learning methods are able to learn formulas in propositional logic which contain n-ary relations (called predicates). These methods are an extension of purely propositional learning approaches which are based on atomic formulas. Thus, an introduction to the sequential covering algorithm for formulas of the latter type [Mit97] is given first.

The aim of the sequential covering algorithm is to learn a set of if-then rules given a set of positive training examples (examples which are classified as true) and negative training examples (examples which are classified as false), where the conditions are represented as a conjunction of literals.

```
SEQUENTIAL-COVERING($Target\_attribute$,$Attributes$,$Examples$,$Threshold$)
  • $Learned\_rules \leftarrow \{\}$
  • $Rule \leftarrow$ LEARN-ONE-RULE($Target\_attribute, Attributes, Examples$)
  • while PERFORMANCE($Rule, Examples$)$> Thresholds$, do
    • $Learned\_rules \leftarrow Learned\_rules \cup \{Rule\}$
    • $Examples \leftarrow Examples -$ {examples correctly classified by $Rule$}
    • Rule $\leftarrow$ LEARN-ONE-RULE($Target\_attribute, Attributes, Examples$)
  • Learned_rules $\leftarrow$ sort $Learned\_rules$ according to PERFORMANCE over $Examples$
  • return $Learned\_rules$
```

Figure 6: Pseudocode for the sequential covering algorithm from [Mit97, p. 276].

The pseudocode for this algorithm is given in Figure 6. This algorithm sequentially learns one rule after another. In each iteration, it determines the best rule according to a given PERFORMANCE function. A high score is obtained for rules which cover as much as possible positive and preferably no negative examples.

The determination of a single rule is described in pseudocode in Figure 7. The conjunction, representing the conditions under which the rule is applicable, is extended iteratively by adding additional attribute value restrictions ($A_i = a_{ij}$) which causes the conjunction to cover less positive and less negative examples. Note that for performance issues only the k best alternatives for such a restriction are selected for further extension. If there are no more attributes left, the rule with the best score is returned. In addition, all positive examples are removed which are covered by this rule. As a result, only the remaining positive examples have to be regarded in the next iterations. If no more positive training examples are left, the algorithm terminates.

One drawback of this method is that only propositional formulas can be learned but not formulas involving n-nary predicates which means that it is not possible for instance to learn the following formula:

$$parent(x, y) \land parent(x, z) \land y \neq z \land male(y) \rightarrow brother(y, z) \tag{10}$$

Two possible approaches exist to extend the algorithm to formulas involving predicates [Dže07] where the second approach is preferable due to its higher efficiency.

The first approach creates propositional atoms for every possible variable assignment for the predicate arguments using all constants appearing in the training data, e.g., for 5 constants and one two-place predicate, $5 \times 2 = 10$ atomic propositions have to be introduced.

```
LEARN-ONE-RULE(Target_attribute, Attributes, Examples, k) Returns a single rule that covers some
of the Examples. Conducts a general to specific greedy beam search for the best rule, guided by the
PERFORMANCE measure.
  • Initialize Best_hypothesis to the most general hypothesis $\emptyset$
  • Initialize Candidate_hypothesis to the set {Best_hypothesis}
  • while Candidate_hypothesis is not empty, do
  1. Generate the next more specific Candidate_hypotheses
      • All_constraints $\leftarrow$ the set of all constraints of the form ($A_i$=$a_{ij}$), where
          a is a member of Attributes, and v is a value of a that
          occurs in the current set of Examples
      • New_candidate_hypotheses $\leftarrow$
          for each h in Candidate_hypotheses,
              for each c in All_constraints,
                  • Create a specialization of h by adding the constraint c
      • Remove from New_candidate_hypotheses any hypotheses that
          are duplicates, inconsistent, or not maximally specific.
  2. Update Best_hypothesis
      • For all h in New_candidate_hypothesis do
          • If (PERFORMANCE(h, Examples, Target_attribute) >
              PERFORMANCE(Best_hypothesis, Examples, Target_attribute))
              then Best_hypothesis $\leftarrow$ h
  3. Update Candidate_hypotheses
      • Candidate_hypotheses $\leftarrow$ the k best members of
          New_candidate_hypotheses, according to the PERFORMANCE measure
  • Return a rule of the form
      "IF Best_hypothesis THEN prediction" where
          prediction is the most frequent value of Target_attribute
          among those Examples that match Best_hypothesis.
```

Figure 7: Pseudocode for the function LEARN-ONE-RULE from [Mit97, p. 278].

The second approach extends the sequential covering algorithm in such a way that in addition to adding a predicate to a conjunction, an equality constraint for two variables appearing in any predicate of the regarded conjunction can be added as well. In the example rule above, equality constraints have to be inserted to identify the sole argument (i.e., y) of the predicate *male* with the second argument of the first occurrence of the predicate *parent*. Analogously, the first argument (i.e., x) of the first occurrence of the predicate *parent* is identified with the first argument of the second occurrence of the same predicate. Furthermore, several aspects concerning the bindings of the variables of the conjunction to the constants of the examples have to be dealt with.

The relational version of the sequential covering algorithm is capable of learning graph substructures. Thus, it can be employed to learn graph patterns for detecting semantic relations (e.g., hyponymy or meronymy) in a semantic network.

3 Knowledge Acquisition

3.1 Learning Textual Entailments

An *entailment* is a logical relation between two formulas such that all models for the first formula (A) are also models for the second (B): $A \models B$. Entailments can be used to represent inferences occurring in natural language. For this application scenario, the term *textual entailment* is commonly used. According to the definition of the *Pascal Textual Entailment Workshop Challenge* [GMDD07], a textual entailment relation holds between a base text and a hypothesis if the validity of the hypotheses can almost certainly be deduced from the validity of the base text. A special case of a textual entailment is a paraphrase, i.e., a restatement which is semantically equivalent to the original phrase.
Example:

1. *Max Frisch is the author of the novel "Stiller".*
2. *Max Frisch wrote the novel "Stiller".*

There are several algorithms to extract paraphrases automatically from texts. At first these methods usually extract so-called templates. Such a template contains several slots which can be filled by certain expressions.
Example: *X is author of Y* $\leftrightarrow$ *X has written Y.*
A template is usually based on surface representations [RH02, BL03], predicate-argument structures [LP01] or subtrees [SSG03].

In the approach of Ravichandran and Hovy, a list of surface-oriented pairs of expressions (called anchors) is given [RH02]. The expressions of each pair are related in a certain way to each other.
Example: The relation *born in* holds between *Mozart* and *1756* since Mozart was born in 1756.
Sentences which contain both related expressions are then extracted by employing a search engine. Afterwards, frequently occurring character strings in these sentences are identified. Templates are then created by replacing the anchors in these character strings (in the example *Mozart* and *1756*) by slot variables. Employing a bootstrapping approach, the list of expression pairs can be extended automatically. This is done in such a way that the identified templates are tried to be applied on all sentences of the text corpus. In case of success, the associated expression pairs are used as anchors for further template extracting.

A disadvantage of this bootstrapping method is that only templates of the same type (e.g., templates for birthdays) can be learned. Hence, to reach a good coverage, anchor pairs of many different types have to be provided. For this reason, this method was extended by the TE/ASE system in such a way that also the anchors can be determined automatically[SDC04]. This approach uses as input only a word list which can be easily extracted from a monolingual dictionary. The words from this list, so-called pivot elements, serve the purpose of extracting anchor pairs from a given text corpus. The pivot elements are characterized by the fact that they are assigned special valency information (i.e., for verbs the information that they require a subject and possibly one or several objects). If the pivot element and its arguments build a collocation, i.e., they occur together in a lot of sentences, these arguments will be used as anchors for determining paraphrases basically analogous to the approach of Ravichandran and Hovy as explained above.

Szepktor et al. raised the recall of the TE/ASE system by 6 % (relatively ca. 10 %) through preprocessing the sentences. The preprocessing consists in changing passive in active voice, in replacing abbreviations by their full written form, and in simplifying appositions and conjunctions [SD07].

The methods described above determine a paraphrase equivalence class by keeping a certain expression pair fixed and letting the substrings between them vary. In contrast, Sekine recognize paraphrases by identifying keywords occurring in character [Sek05] strings between two named entities (e.g., the word *unit* for the phrase *the unit of*). These keywords identify the most important, i.e., semantically relevant part of these expressions. The sentences of the investigated text corpus are clustered according to these keywords. Inside of a cluster are mostly text passages which are paraphrases of each other. This method is additionally combined with a variant of the approach of Ravichandran and Hovy as described above.

Lin and Pantel describe an approach to determine an equivalence class where the expression pair is varied too. The approach of Lin and Pantel extracts all paths from a dependency tree which connect two nouns with each other [LP01]. These nouns are referred to as slot fillers. Two text passages are considered as similar (and therefore as paraphrases) if the associated paths connect essentially the same nouns with the same frequency with each other. For this purpose, every path is assigned two vectors which specify the number of occurrences of the slot filler expressions in the text corpus. For the determination of the similarity of the vector pairs a distance function is defined, where the occurrence of usually rare nouns has more influence for the similarity calculation than of more frequent ones. Note that this algorithm is quite similar to the synonymy/hyponymy recognition by textual context analysis [CPSTS05].

The methods described so far determine paraphrases typically on the non-sentence phrase level. In contrast, the approach of Barzilay et al. determines paraphrases on the sentence level [BL03]. For that, the sentences of two different text corpora (for instance Barzilay et al. used different newspaper corpora) are compared with each other. A sentence pair is considered to contain paraphrase if the sentences have a similar syntactic structure and contain words with identical meaning (n-gram-Overlap). In principle, this approach is similar to a lot of shallow methods for the determination of textual entailment [KM06, IKN06, Ada06]. An advantage of this approach is that very complex patterns can be recognized, but it should be kept in mind that the learned paraphrases might be very specific nevertheless.

Bilingual corpora are also employed for paraphrase recognition [BM01]. The approach of Barzilay et al. is based on the assumption that two independent translators create translations with identical contents but different words, i.e., the two translations should contain many paraphrases. Barzilay et al. first align the sentences from both corpora by identifying sentence pairs containing a lot of identical words. Afterwards, text fragments with identical words are determined for each aligned sentence pair. Paraphrases are then expected to appear between these text fragments.

Zhao et al. focus on a special kind of paraphrases, i.e., paraphrases of questions [ZZL07]. For this purpose, they analyze the logs for the online encyclopedia Encarta[1] which allows the formulation of questions in natural language. In the first step, questions are clustered according to their type, e.g., *abbreviation*, *animal*, *body*, *color*, *creative*, *explanation*, *definition*, etc. Afterwards, the questions inside such a cluster are further subdivided into other clusters according to the content words. A question is usually part of several of those subclusters. Each question is compared to every other question of the same subcluster. Whether the elements of a question pair are

[1]URL: `http://de.encarta.msn.com/`

paraphrases of each other was learned by means of PAUM (perceptron algorithm with uneven margins) [LZHST02]. The features exploited for learning included the following:

- cosine similarity feature between content words.
- named entity overlapping feature (NEF): Overlapping rate of named entities, computed by a variant of the overlap index (see Section 3.3).
- user click preference: Comparison of mouse clicks which assumes that questions leading to similar clicks are similar.
- WordNet synonym feature: Overlapping rate between synonyms.

All of the methods for extracting textual entailments described so far are limited to paraphrases. However, not all kinds of textual entailment text pairs are actually considered as paraphrases. For instance the action of *bestowing* implies the action of *giving* but not the other way around. A method capable of learning textual entailments which are not paraphrases was developed by Zanzotto and Moschitti [ZM06]. As training data they use text pairs from the *Pascal Textual Entailment Challenge*[GMDD07] for which a textual entailment relation is known to hold. They investigate in what way the syntactic structure of the base text can be transformed into the syntactic structure of the hypothesis. Textual entailment rules are then extracted which describe these transformations.

Moreover, special textual entailments can be learned by using the algorithm described in Section 3.7 which extracts useful knowledge from generic sentences.

Except from texts, entailments are also often learned from lexicon glosses which contain verbal explanations for lexical entries.
Example: *boxing*: *to hit someone with the fist.*
The approach of Glöckner et al. [GHO05] parses the explanation of a word in GermaNet with a deep analyzer and stores the resulting semantic representation in the knowledge base. The main problem Glöckner et al. were confronted with is to disambiguate the words occurring in such an explanation since the available textual context is usually quite small.

Putting things together, the most existing approaches are based on a surface representation or on a syntactic structure, which leads to several shortcomings in comparison with a deep semantic representation. There exist many ways to express semantically identical facts using syntactic or surface representations, which leads to a high number of templates learned. Thus, by using a semantic representation, the amount of learned entailment templates can be reduced considerably. Moreover, the learned entailments are also more generally usable. Therefore, the usage of a semantic representation can diminish the resource consumption by additionally improving the coverage. A normalization or preprocessing step is superfluous or can at least be considerably diminished. Furthermore, semantic representations are always based on concepts instead of word forms, which can also reduce problems connected with ambiguous words, which of course depends on the quality of the applied Word Sense Disambiguation. Finally, the type of entailments which can automatically be learned by the approaches described above is quite limited. The methods usually concentrate on certain types of textual entailments (e.g., paraphrases).

3.2 Learning Ontological Sorts and Semantic Features

Ontological sorts are an important part of the multi layered extended semantic network (MultiNet) formalism [Hel06]. MultiNet is a knowledge representation formalism for the semantics of natural language. Ontological sorts build a basic ontology of concepts. Each concept in a semantic lexicon is assigned to one ontological sort. in a given sortal hierarchy. The top level sort is named *Entity* and subsumes all other sorts. On the next level, the following sorts are defined:

1. Objects; associated grammatical category: noun phrase
   - Concrete objects: e.g., *milk*, *honey*
   - Abstract objects: e.g., *race*, *robbery*
2. Situations; associated grammatical category: verb phrase
   - Static situations, e.g., *being pale with hunger*, *having a temperature of...*
   - Dynamic situations, e.g., *work*, *rain*
3. Situational Descriptors; associated grammatical category: prepositional phrase
   - Temporal situational descriptors, e.g., *yesterday*, *on Mondays*
   - Local situational descriptors, e.g., *on the roof*, *under the table*
   - Modalities, e.g., *probably*, *necessary*
4. Qualities; associated grammatical category: adjectival phrase
   - Properties in the narrower sense, e.g., *tall*, *heavy*
   - Relational qualities, e.g., *equivalent*, *inverse*
   - Functional qualities, e.g., *philosophical*, *chemical*
5. Quantities; associated grammatical categories: noun phrase, adjectival phrase
   - Quantificators, e.g., *all*, *more than the half*
   - Unit of measurement and measurements, e.g., *3 kg*, *a few meters*
6. Graduators; associated grammatical category: adverbial phrase
   - Qualitative graduators, e.g., *especially*, *rather*
   - Quantitative graduators, e.g., *almost*, *nearly*
7. Formal entities: objects which do not have a direct linguistic representation, e.g., *in the upper half of the figure*, *the components in the left corner*

Currently, more than 40 different ontological sorts are defined in MultiNet. Additionally, the MultiNet formalism defines several semantic features, i.e., semantic properties of objects which are either be present or not present. Note that the value of a semantic feature can also be undefined.
Examples of semantic features are *animate* (*living being*) and *human* (*human being*). Ontological sorts and features can be employed for ontology building and hyponymy detection. A hyponymy relation between two entities normally requires the sorts of the hypernym and of the hyponym to be identical. Furthermore, the hyponym has to contain all semantic features of the hypernym.

Socher et al. describe a method to derive ontological sorts and semantic features for nouns by text mining [SBO07]. In the first step they build complex semantic sorts consisting of an ontological sort and a list of semantic features where the latter are set appropriately corresponding to the associated ontological sort. The hierarchy of the complex semantic sorts is defined by the hierarchy of the associated ontological sorts.

The aim of this algorithm is to derive a complex semantic sort for a noun by examining the associated verb with that noun in a deep subject/object position or the modifying adjective in case such a verb or adjective exists. This procedure is based on the fact that an adjective or verb usually occurs in texts together with nouns having similar ontological sorts or semantic features. Consider for instance the adjective *charming*. The modified noun should usually be assigned the semantic feature *human*.

This approach determines for every noun the ontological sort and semantic features with the highest probability by investigating the associated verb or modifying adjectives by means of a frequency analysis. Using this information, this approach looks for the most specific complex semantic sort in the sort hierarchy which is compatible with the extracted sort and features. The ontological sorts and semantic features are then directly obtained by the definition of that complex semantic sort. By learning complex semantic sorts instead of simple ontological sorts and semantic features directly, inconsistencies between sorts and features can be avoided. In the evaluation this approach showed a high precision but a rather low recall.

3.3 Learning Synonyms and Near-Synonyms

In wide areas of NLP, e.g., for question answering systems, information retrieval, and recognizing textual entailment it is important to decide whether concepts are semantically similar or not. Note that almost all approaches for detecting semantic similarity are shallow and deal with words instead with semantic concepts. Thus, in the following, only the determination of the semantic similarity between *words* is described. Analogously, this basically holds for learning the other semantic relations described in this work as well, i.e., hyponymy, meronymy and antonymy.

The similarity is usually given by a numerical similarity measure. One way to define such a measure is to derive for both words a vector representation from their textual contexts and compare both vectors with each other. Such approaches are based on the assumption that words are semantically similar if they appear in equivalent (or very similar) textual contexts [MS99, Har68]. The textual context of a word is given by neighboring words, or words which are related by a certain grammatical relation to the investigated word (head modifier in [Rug97] or predicate argument structures in [Hin90]).

The word context can be represented by vectors. The element v_i of the vector for a word w determines the frequency of another word u_i appearing in the textual context of w. Some approaches determine only binary values, i.e., $v_i = 1$ if the number of common occurrences is above a given threshold. By using binary values a set representation can easily be derived. A word u_i belongs to the set X of the words occurring with w if and only if $v_i = 1$.

The semantic similarity between two words w and u can then be approximated by the comparison of the associated sets X and Y. Common similarity measures are given in Table 1. Most of these formulas reach the maximum value of 1 only if both sets are identically. One exception to this rule is the overlap index, where the maximum value is reached if one of the sets is a subset of the other.

Table 1: Similarity measures for binary vectors from Manning/Schütze.

Similarity measure	Definition
Matching coefficient	$\|X \cap Y\|$
Dice coefficient	$\frac{2\|X \cap Y\|}{\|X\|+\|Y\|}$
Jaccard/Tanimoto coefficient	$\frac{\|X \cap Y\|}{\|X \cup Y\|}$
Overlap coefficient	$\frac{\|X \cap Y\|}{min\{\|X\|,\|Y\|\}}$
Cosine	$\frac{\|X \cap Y\|}{\sqrt{\|X\|\|Y\|}}$

Table 2: Similarity measures for continuous values, $\mathbf{x} = P(c|w_1)$, $\mathbf{y} = P(c|w_2)$, $F(w) = \{c|I(c,w) > h\}$, where h is some given positive threshold.

Similarity measure	Definition
Cosine	$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\lvert\mathbf{x}\rvert\lvert\mathbf{y}\rvert}$
p-norm	$\Vert\mathbf{a}\Vert_p = (\lvert a_1\rvert^p + \ldots + \lvert a_n\rvert^p)^{1/p}$
KL divergence	$D(\mathbf{x}\Vert\mathbf{y}) = \sum_{i=1}^{n} x_i \log_2 \frac{x_i}{y_i}$
α-skew	$D(\mathbf{x}\Vert(\alpha\mathbf{y} + (1-\alpha)\mathbf{x}))$
Information radius	$D(\mathbf{x}\Vert\frac{\mathbf{x}+\mathbf{y}}{2}) + D(\mathbf{y}\Vert\frac{\mathbf{x}+\mathbf{y}}{2})$
Precision	$sim_p(w_2, w_1) = \frac{\sum_{c \in F(w_1) \cap F(w_2)} I(c,w_2)}{\sum_{c \in F(w_2)} I(c,w_2)}$
Recall	$sim_r(w_2, w_1) = \frac{\sum_{c \in F(w_1) \cap F(w_2)} I(c,w_1)}{\sum_{c \in F(w_1)} I(c,w_1)}$
Harmonic Mean	$sim_{hm}(w_2, w_1) = \frac{2 sim_p(w_2,w_1) sim_r(w_2,w_1)}{sim_p(w_2,w_1) + sim_r(w_2,w_1)}$

One drawback of the approaches with binary values is that the similarity measures might represent the actual semantic similarity only very roughly. This can be improved by using measures based on continuous (i.e., non-binary) values. Such measures often require the compared vectors to be normalized. In this case, the vector components can also be interpreted as the probability $x := p(c|w)$ that a word w has the textual context c.

In the following several continuous similarity measures are introduced which are all formally specified in Table 2. The cosine method for binary vector components can be generalized to the continuous case as well. Another frequently used measure is the so-called p-norm where the p-norm is applied on the vector difference of both vectors. In practice, the 1- and 2-norm are most often used [DLP97, GM98]. In case of normalized vectors, ranking word pairs according to the 2-norm or the cosine norm leads to identical results.

A more probability-oriented approach is the Kullback-Leibler(KL) divergence $D(\mathbf{x}||\mathbf{y})$ [KL51]. A drawback of this approach is that a summand is not defined (or infinite if a limit approach is used) in the case that one component of $\mathbf{y}$ is zero. This problem can be overcome by employing the α-skewed variant [Lee99] and setting α to a high value near one like 0.99. A further drawback of the KL divergence is that it is not symmetric, i.e., $D(\mathbf{x}||\mathbf{y}) \neq D(\mathbf{y}||\mathbf{x})$. Therefore, this measure was modified by Dagan et al. to the so-called information radius [DLP97]. Dagan et al. compared information radius, 1-Norm and KL divergence with each other for estimating the probability for the occurrence of word pairs (w_1, w_2) which do not show up in the training set. It was investigated how often words which are semantically similar to w_1 appear together with w_2 in a given text corpus. The semantic similarity was calculated with all three approaches alternatively. The evaluation showed that the information radius approach led to significantly lower errors than the other methods.

The similarity measures *precision*, *recall*, and *harmonic mean* as described in Table 2 [WWM04] are based on the pointwise mutual information, which is a measure of the association between a word and a given context (e.g., neighboring word) which is defined as:

$$I(c, w) = \log_2 \frac{p(c|w)}{p(c)} \tag{11}$$

Semantic similarity can be employed to determine synonymy relations between word pairs. Note, however, that usually a high similarity is also obtained if a hyponymy or an antonymy relationship holds. Hence, a combination with other methods is necessary for getting an acceptable precision.

An alternative approach for synonymy extraction by context analysis was proposed by Biemann et al. [BBQ03, BBQ04]. This approach takes the possibility into account that the knowledge base already contains synonyms and also handles the merging of already existing with newly derived synsets (synset: synonymy equivalence class). The key feature of this approach consists in the fact that *iterated* co-occurrences are used. The first order co-occurrence set of a word is defined as the set of all words which appear in its textual context. A second order co-occurrence set of a word is then the set of all words which appear together with this word in a first order co-occurrence set. In principle, this process can be repeated infinitely to build co-occurrence sets of third, fourth, etc. order.

After the co-occurrence sets are determined, they have to be merged with the existing synsets. This is done by determining for each existing synset the most similar co-occurrence sets, i.e., the sets having the largest intersection with that synset. Elements occurring in all of these co-occurrence sets are then added to the synset. How many co-occurrence sets are used depends on the so-called semantic homogeneity of the sets which is measured by certain semantic properties.

Another approach exploiting co-occurrences was introduced by Edmonds [Edm97]. This approach creates a lexical co-occurrence network, where an arc exists between two words if both words co-occur in a given text corpus. The significance score of two words w_1 and w_1 for being synonymous is then defined as:

$$sig(w_1, w_2) = \frac{1}{d^3} \sum_{q_i \in Q(w_1,w_2)} \frac{t(q_{i-1}, q_i)}{i} \tag{12}$$

where:

- $Q(w_1, w_2)$: the shortest path $q = (q_1, \ldots, q_d)$ in the co-occurrence network from w_1 to w_2
- d: the length of the shortest path
- $t(q_{i-1}, q_i)$: value for the statistical t-test that the occurrences of q_{i-1} and q_i are independent (see [CGH+94])

The significance score reaches high values if the path between both words is rather short and the t-value of the path components (which measures co-occurrence) is quite large. Edmonds describes an application scenario of this method in the area of text generation. Let us assume, there exist several synonymous words to fill in a gap in a sentence. That word should then be chosen which fits best into the textual context, i.e., to the other words of the sentence. The best fitting word w_b for a sentence S is then given as:

$$w_b = arg \max_{w'} \sum_{w \in S} sig(w', w) \tag{13}$$

3.4 Learning Hypernyms and Hyponyms

According to Fromkin et al., hyponyms are a set of related words whose meanings are specific instances of a more general word [FRH02].
Example: A **hammer** is a **tool**.
In this example, *hammer* is called a hyponym of *tool* and *tool* is a hypernym of *hammer*. The knowledge about hyponymy relationships is important in many areas of NLP. Consider for example the following Textual Entailment problem:
Base text: *Mr. Peters bought a Mercedes.*
Hypothesis: *Mr. Peters bought a car.*
If it is known that *Mercedes* is a hyponym of *car*, the hypothesis can be deduced from the base text.

Automatic hypernym extraction is usually based on text mining in large corpora, text mining in dictionaries (mono and bilingual), or document clustering.

Let us first take a closer look of extracting hyponyms via text mining in large corpora. Basically, there are two methods to do this:

1. Analyzing the syntagmatic relations of sentences. Approaches following this principle often use text patterns for hyponymy detection.
2. Analyzing the paradigmatic relations of sentences. This method is based on the fact that a hypernym often occurs in the same textual contexts as its hyponyms.

The approach of Hearst belongs to the first kind of algorithms. It employs the following patterns for the recognition of hyponymy relations between nouns in English texts [Hea92] ($hyponym(a_1, a_2)$ means: a_1 is hyponym of a_2):

- such NP_0 as $\{NP_{2,\ldots,n}\}^*$ (or|and) NP_1
  $\Rightarrow hyponym(NP_i, NP_0)\ i \in \{1, \ldots, n\}$
  Example: *works by such authors as Herrick, Goldsmith and Shakespeare.*
- $NP_1\{, NP_{2,\ldots,n}\}^*$ or other NP_0
  $\Rightarrow hyponym(NP_i, NP_0)\ i \in \{1, \ldots, n\}$
  Example: *Bruises, wounds, broken bones or other injuries*
- $NP_1\{, NP_{2,\ldots,n}\}^*$ and other NP_0
  $\Rightarrow hyponym(NP_i, NP_o)\ i \in \{1, \ldots, n\}$
  Example: *. . . temples, treasuries, and other important civic buildings.*
- $NP_0, including\{NP_{2,\ldots,n}\}^*$ and/or NP_1
  $\Rightarrow hyponym(NP_i, NP_0)\ i \in \{1, \ldots, n\}$
  Example: *All common-law countries, including Canada and England . . .*
- $NP_0, especially\{NP_{2,\ldots,n}\}^*$and/or NP_1
  $\Rightarrow hyponym(NP_i, NP_0)\ i \in \{1, \ldots, n\}$
  Example: *most European countries, especially France, England and Spain . . .*

These patterns are applied to a large text corpus and hyponyms/hypernyms are extracted according to the associated rules. One critical point of this algorithm is the size of the text corpus. If it is quite small, the patterns will possibly be matched too rarely. In this case, the number of extracted hyponymy relations can be rather small too.

Therefore, Cimiano et al. extended this approach in such a way that they use the entire Web as corpus and generate search engine queries by inserting a pair of arbitrary terms t_1 and t_2 (a term is considered to be some arbitrary noun phrase) in the hypernym and hyponym position position of the pattern [CPSTS05]. The probability that a hyponym relation holds between two terms t_1, t_2 is then approximated by the relative frequency of search results found divided by the number of patterns found for t_1.

Apart from the usage of handcrafted patterns there was also some work to determine text patterns automatically [SJN05]. For that, Snow et al. collected sentences in a text corpus with known noun pairs of hyponyms and hypernyms. 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 hypernym 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 occurring 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. [MJ04]. 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 similaraty cluster. The patterns to be used for hyponymy detection are the candidate patterns of all clusters found.

One problem that arises by the usage of patterns is that the patterns often have either a high precision or a high recall for detecting hyponyms but not both. Thus, most approaches focus on patterns with a high precision in order to avoid potential errors. In contrast, the approach of Pantel et. al, called ESPRESSO, uses additionally patterns with high recall and low precision, so-called generic patterns [PP06]. Note that ESPRESSO is capable of determining semantic relations in general, not only hypernymy relations. To compensate for the smaller reliability of generic patterns a term pair $t_i := (t_{i_1}, t_{i_2})$ extracted by a certain pattern is only classified to a semantic relation if the confidence value associated to this word pair exceeds a certain threshold. This confidence value is given as:

$$S(i) = \sum_{p \in P_R} S_p(i) \cdot \frac{r(p)}{T} \tag{14}$$

with:

- P_R: pattern classified as being reliable
- $S_p(i)$: pointwise mutual information between a pattern p and a term pair $t_i = (t_{i_1}, t_{i_2})$
- $r(p)$: reliability of a pattern p

The pointwise mutual information is given as:

$$pmi(i,p) = \log_2 \frac{|t_{i_1}, p, t_{i_2}|}{|t_{i_1}, *, t_{i_2}| \cdot |*, p, *|} \tag{15}$$

where $|t_{i_1}, p, t_{i_2}|$ denotes the number of matches of pattern p in a text corpus where the slots of the pattern are instantiated to t_{i_1} and t_{i_2} (“*” is the wildcard operator which matches to any expression).

The reliability of a pattern is determined by analyzing the mutual information between the applicability of this pattern and occurrences of word pairs of a certain semantic relation. A high mutual information expresses a strong association between a pattern and a word pair.

Gliozzo describes a syntagmatic approach to extract hyponymy relations, not employing text patterns [Gli06]. Instead, his approach is based on the fact that in many cases, the semantic relation (e.g. hyponymy or meronymy) between the subject and the object of a sentence is expressed by the verb.
Examples:
*A hammer **is** a tool. Hammer* is a hyponym of *tool.*
*A book **contains** several pages. Page* is a meronym of *book.*
He assumes that a relation, indicated by a verb v, between a subject w_1 and an object w_2 of a sentence is more likely to occur, if all three expressions (subject, object

and verb) build a collocation. Thus, the confidence score that a certain semantic relation is expressed is given by:

$$score(w_1, v, w_2) = \frac{\frac{F(w_1,v,w_2)}{\min\{F(w_1,*,*),F(*,*,w_2)\}}}{\frac{F(w_1,v,*)}{F(w_1,*,*)} + \frac{F(*,v,w_2)}{F(*,*,w_2)}} \tag{16}$$

where $F(w_1, v, w_2)$ counts the number of occurrences of the triple (w_1, v, w_2) in any sentence with w_1 being the subject, v the verb, and w_2 the object (the expression $*$ can be matched with any term). These triples are retrieved by regular expression matching over the output of a shallow parser. Note that an evaluation of this method was missing.

Apart from its syntagmatic approach, Cimiano et al. also proposed a paradigmatic method [CPSTS05]. This method examines the textual context of potential hypernym/hyponym pairs in a large text corpus. This approach is based on the observation that the textual surrounding of a hypernym can also occur with the corresponding hyponym but not necessarily the other way round. Let us consider an example for this fact with the hypernym *flower* and the hyponym *rose*. Let us assume that the regarded context just contains the preceding word. *Prickly rose* is a collocation but not *prickly flower*. However, both *nice rose* and *nice flower* are collocations.

The probability that a hyponymy relation holds between two terms is estimated by:

$$hyponym(t_2, t_1) = \frac{|features(t_1) \cap features(t_2)|}{|features(t_1)|} \tag{17}$$

The used features are the occurrences of modifying adjectives, prepositions, possessive pronouns and certain sentence constructions. Moreover, Cimiano et al. extract hyponym relations from multi-word nominal phrases. If such a noun phrase p_1 consists of an adjective followed by another nominal phrase p_2, then p_1 is often the hyponym of p_2, e.g., the nominal phrase *international conference* is a hyponym of *conference*. To improve the accuracy of the results, Cimiano et al. combine several approaches for hyponym recognition using alternatively naïve Bayes, perceptron, and decision tree classifiers.

An approach which does not rely on context analysis or pattern matching is described by Kashyap et al. This approach constructs a taxonomy by document clustering [KRTS05]. All documents to be used for building the taxonomy are represented by a matrix M where the component M_{ik} specifies the relative frequency of term t_k occurring in document d_i. The dimension of this matrix is reduced using the same methods as with Latent Semantics Analysis (LSA) [LFL98].[2] All documents are clustered hierarchically according to their LSA-transformed frequency vectors. Afterwards certain cluster nodes are selected to build the taxonomy where the number of the selected nodes can be influenced by a parameter. The associated linguistic label of a single node of the taxonomy is determined according to the term occurrence vectors belonging to the documents in the associated cluster. In particular, the term having the smallest distance to the centroid of the cluster is chosen for labeling. The hierarchy of the taxonomy follows directly from the cluster hierarchy.

An alternative approach for document clustering is proposed by Thanh Tho Quan et al. and is based on fuzzy set theory and formal concept analysis [QHFC04].

[2]An alternative method without LSA was also evaluated but lead to inferior results.

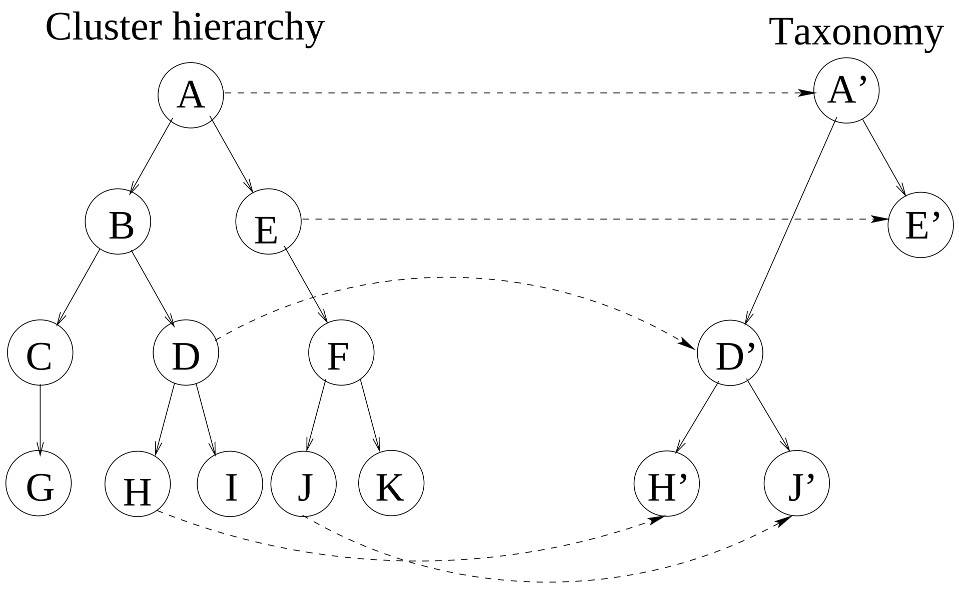


Figure 8: Building a Taxonomy via text mining. Some of the cluster nodes are selected for the construction of the taxonomy.

Table 3: Example for a fuzzy formal concept analysis with document set $G = \{d_1, d_2, d_3\}$ and term set $M = \{D, C, F\}$.

	D	C	F
d_1	0.8	0.12	0.61
d_2	0.9	0.85	0.13
d_3	0.1	0.14	0.87

A fuzzy set $\varphi(G)$ over a set G is a set in which each element g is assigned a certain degree of membership $\mu(g)$, usually ranging from zero to one. The cardinality of this set is given as the sum of all membership degrees: $|\varphi(G)| = \sum_{g \in G} \mu(g)$.

In a first step Quan et al. construct a fuzzy set $I = \varphi(G \times M)$ where G denotes a set of documents and M a set of given terms, which are possibly relevant for these documents. This fuzzy set can be represented as a matrix where each entry I_{ij} ranges from zero to one and specifies the relevance (membership) of a term t_j for document d_i. An example is shown in Table 3 with $G = \{d_1, d_2, d_3\}$ and $M = \{C(lustering), D(ataMining), F(uzzy)\}$. Membership values which are smaller than a given threshold are ignored and therefore removed from the fuzzy set.

In the next step, each document is assigned a membership value by building the minimum value of all the concept membership values which are associated with that document. Thus, for document d_i this value is the minimum value of all elements of row i of the matrix I which are not less than h.

A concept is defined as a pair $(\varphi(G'), M')$ of documents $\varphi(G')$ and terms M' with $G' \subseteq G$, $M' \subseteq M$ where G' is the set of all documents in which all terms of M' appear (i.e., membership degree above threshold). A concept is created for every possible subset of terms for which the document set is not empty. Additionally, this set always contains the pair $(G, \{\})$. Consider for example the fuzzy set given in Table 3. If the threshold h is 0.5, no concept is created for the term set {C,F} since the membership value of this term set falls below that threshold for all given documents.

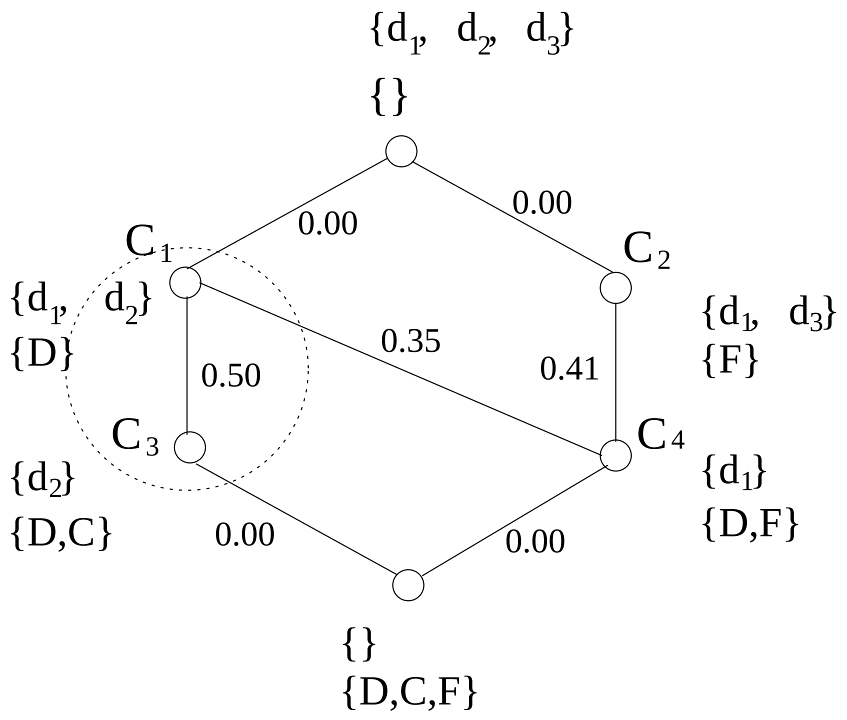


Figure 9: Example of a concept hierarchy from Quan et al. Concept nodes contain tuples of documents and terms. Arcs between concepts denote concept subsumptions and are labeled with the similarity values of the associated concepts.

An ordering is defined on the concept set in such a way that a concept C_1 is considered smaller than a concept C_2 if the associated set of terms of C_1 is subsumed by C_2. The concept ordering for the example from Table 3 is illustrated in Figure 9. Furthermore, a similarity measure is defined on two concepts by the Jaccard/Tanimoto coefficient (see Section 3.3): $\frac{|X \cap Y|}{|X \cup Y|}$.
Example for concepts C_2 and C_4:

$$sim(C_2, C_4) = \frac{|\varphi(\{d_1\})|}{|\varphi(\{d_1, d_3\})|} = \frac{0.61}{0.61 + 0.87} = 0.41 \qquad (18)$$

All concepts which are very similar to each other (i.e., similarity measure above a certain threshold, e.g., 0.5) are combined into a single compound concept. The resulting hierarchy represents a taxonomy. Currently, concepts of this taxonomy are not yet lexicalized, but the automatic assignment of word labels to taxonomy nodes is planned for future work.

Aside from ordinary texts, hypernym relationships are also extracted from dictionaries [Cla98]. Dictionaries are differentiated into mono- and bilingual. A bilingual dictionary contains translations for each word in another language while a monolingual dictionary describes the meaning of a word with a short text passage in the same language. Such a definition often contains a hypernym of the defined word which is exploited by the approach of Rigau.
Example from Miller [CF98, p.24]: ***bird:*** *warm-blooded egg-laying* ***animal*** *having feathers and forelimbs modified as wings*
The hyponym is expected to be the so-called *genus term* (most important term) of the noun/verb definition which is often the first noun occurring in the definition. Rigau employs a special grammar to extract this term. Afterwards, a Word Sense

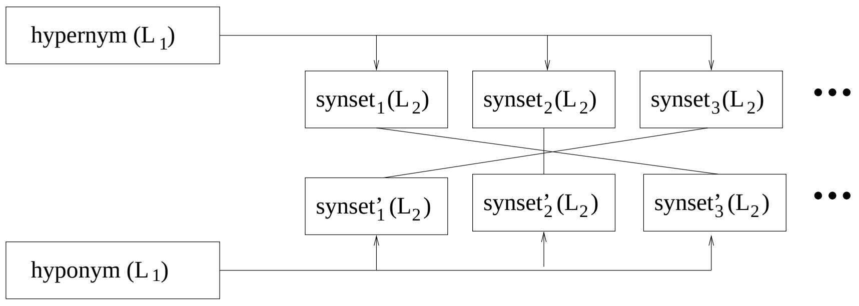


Figure 10: Extracting hyponyms by exploiting a bilingual dictionary. Hyponym and hypernym of language L_1 are both translated to synsets in language L_2.

Disambiguation is applied to the genus term and its textual context to determine the semantic concept associated with that term. Note that the genus term might be a synonym instead of a hypernym of the defined word which can result in circles in the learned taxonomy, e.g., if the definition of the word *car* contains the word *auto* and the other way around. Thus, these circles are identified and the associated words are considered as synonyms.

After the taxonomies for the selected languages (with Rigau: English and Spanish) are fully constructed, Rigau uses a bilingual dictionary to complete information being present in one language but missing in the other. For that, all synsets of language L_2 are identified which are either the translations of the hypernym or the hyponym in language L_1 (see Figure 10). On this basis, the correct synset pair has to be found which represents the hyponymy relation in language L_2. More specifically, the pair is chosen for which the associated synsets are most similar to each other measured by a WordNet based similarity measure. This measure basically reflects the minimum number of WordNet arcs which are required to reach a member of the hypernym synset from a member of the hyponym synset where a lower number of arcs denotes a higher similarity. Note that both synsets can be connected in this graph (e.g., by using the root node) even if no hyponymy relation exists between them yet. Die Bestimmung des Genusterms war in 97% der F"alle korrekt. Die Akkuratheit des Verfahrens ist 80%.

3.5 Learning Meronyms and Holonyms

Meronymy relations can be divided into the following subrelations [WCH87]:

- Component-integral: a relation between an object and one of its components. Important for this relation is the fact that object and component can be perceived separately from each other. For instance, it is possible to clearly distinguish a wheel from a car.
- Member-collection: This relation represents the membership in a set.
  Example: *A soccer player is member of a soccer team.*
- Portion-mass: relations which refer to mass units and their parts.
  Examples: *A meter is part of a kilometer. A slice of the cake is part of the cake.*
- Stuff-object: This relation represents the chemical composition of an object.
  Examples: *Alcohol is part of wine. Carbon dioxide is part of the air.*
- Feature-activity: Activities can usually be divided into several subtasks.
  Example: The following subtasks belong to the activity *going out for dinner*: *Visiting a restaurant*, *Ordering*, *eating* and *payment.*

Table 4: Some of the Patterns Suggested for the Recognition of Meronyms by Girju et al.; (indices below: 1=meronym, 2=holonym).

Number	Pattern	Example
1	NP_1 is part of NP_2	the **engine** is part of the **car**
2	NP_2's NP_1	**girl**'s mouth
3	NP_1 of NP_2	**eyes** of the **baby**, **doors** of **cars**
4	NP_2 verb:have NP_1	The **table** has four **legs**.
5	NP_2 P NP_1	A **bird** without **wings** cannot fly.
6	NP_1 P NP_2	A **room** in the **house**.
7	$NP(N_2N_1)$ (noun compound)	**door knob**
8	$NP(N_1N_2)$ (noun compound)	**turkey pie**

- Place-area: This relation holds between two objects if one of these objects is geographically part of the other object.
  Example: *Germany is part of Europe.*

For the extraction of meronyms, a pattern-based approach is suggested by Girju et al.[GBM06] (see Table 4 for a subset of these patterns).

Constituents which match to these patterns are often but not always related in a meronymy relation. Note that there are big differences regarding the accuracy connected with these pattern. By using the first pattern (*... is part of ...*), meronyms can be recognized very reliably. However, this is not the case for most of the other patterns. For example, the third pattern matches to the sentence "*The son of Pete is lazy.*" although there exists no meronymy relationship between Pete and his son.

In order to find meronymy relations with ambiguous patterns too, a more sophisticated approach is needed. For that, Girju et al. employ the information from the WordNet concept hierarchy [Fel98] of the matched expressions. In a lot of cases the concept taxonomy is an important information for the detection of meronyms. For instance, if both concepts are hyponyms of *human being* then the existence of a meronymy relation can be rejected. However, if both objects are artefacts, then a meronym relation between them is quite possible.

For the classification of the concept pairs (c_1, c_2) which are matched by a pattern, a decision tree approach is used. An example should be classified to *yes* by the tree if a meronymy relation is actually present, to *no* otherwise. A single example consists of a tuple $(s_1, s_2, yes/no)$. At the beginning, s_1 and s_2 are the most general concepts (in the hypernym hierarchy) of c_1 and c_2. These examples are used to train a decision tree. All examples for which no unique decision could be made are collected. In these examples the hypernyms are further specialized which means they are replaced by the next more specific concepts which subsume c_1 (c_2 respectively). To do this, the associations between examples and word pairs have to be remembered. The decision tree is then reconstructed with the modified examples. This process is iterated until all examples are classified correctly or the most specific concepts are reached. Note that this approach is based on concepts not on words. A Word Sense Disambiguation is used to determine the correct concept for each word. This method was evaluated showing a recall of 84 % and a precision of 79 %.

The approach of Costello [Cos07] also employs the concept taxonomy for meronymy detection. In contrast to Girju et al., he focuses only on noun-noun compounds and his approach is based on words instead of semantic concepts. Note that Costello does not concentrate on meronymy relations only but extract a whole range of semantic relations .

During the training phase, he tags a large number of word pairs appearing in noun-noun compounds with their semantic relations. To determine the semantic relation for a word pair of a previously unseen compound, all elements of the Cartesian product of the senses and hypernym senses of its (two) components are investigated, i.e., if the first component has n (hypernym) senses and the second component m, there will be a total of nm of such pairs. The semantic relation is chosen which most often holds between any sense pairs of the Cartesian product as determined by the initial tagging.

A further important characteristic which is employed for meronymy recognition is the physical size of an object since a meronym is always smaller than the associated holonym [AIMO07]. The approach of Aramaki et al. determines the size of an object employing a search engine query with several patterns. The size information is combined, via a support vector machine, with the extraction of meronyms employing two kinds of text patterns. One of the most serious problems, Aramaki et al. were confronted with, consists in the fact that the sizes can vary quite strongly. For instance, the word *car* can also denote a *model car* which is much smaller than a real car.

3.6 Learning Antonyms

Antonyms are important for many tasks in the area of NLP. Consider for example the following Textual Entailment problem:
Base text: *The car of Dr. Peters is slow.*
Hypothesis: *The car of Dr. Peters is fast.*
If it is known that *fast* is an antonym to *slow*, it can be recognized that the hypothesis cannot be inferred from the base text. In general, antonyms are a typical indicator that certain entailments do not hold.

The approach of Lucero et al. [LPJS04] recognizes antonyms by a combination of the following three methods:

- Antonyms are identified by the text patterns[3]:
  - anto_1 * but * anto_2
  - anto_1 * but rather * anto_2
  - from * anto_1 to * anto_2
  - anto_1 and * anto_2
  - anto_1 or * anto_2

  For instance, a pattern "*anto_1*but*anto_2" can be employed to find antonyms from a sentence like: *The song was not only not bad but even good.*
- Distance between both antonyms: Antonyms often appear quite close to each other in a sentence (see example above).
- Check for synonymy: The approach of Edmonds (see page 19) is used to estimate the probability that both words are synonyms and not antonyms.

The total score of a word pair being related in an antonymy relation is given by the sum of three single scores reflecting the three methods as described above.

[3]the text patterns are translated from Spanish to English

3.7 Learning Other Kinds of Common Sense Knowledge

There exists a lot of knowledge which cannot be extracted with the approaches described above, e.g., typical properties of objects. In the following we introduce two approaches which do not restrict the type of common sense knowledge to acquire.

A method which learns common sense knowledge via text mining was proposed by Suh et al. [SHK06]. This approach extracts all statements which are contained in generic sentences, like *Bees produce honey.* This is based on the fact that generic sentences usually contain common sense knowledge. Whether a sentence is meant generically or not is determined using several heuristics (e.g., missing article or given date or time). Note that usually only a small portion of sentences in a corpus is generic which means that the recall of this method is quite low.

In contrast, the approach of [HLQW01] extracts common sense knowledge from non-generic sentences as well. In particular, it determines triples containing two noun phrases and a predicate where the latter specifies the kind of relationship between the noun phrases. This algorithm assumes that two elements of this triple are already known and the third one is to be determined by corpus analysis. For that, all collocation terms of the two given expressions are determined separately by employing the statistical G-test (see [Dun93]). The candidate set for the third expression is given as the intersection of both collocation sets.

3.8 Learning Other Kinds of Knowledge

There exist several kinds of knowledge which can be contained in a knowledge base or a semantic lexicon which can also be learned automatically but are not in the primary focus of this report. This includes but is not limited to named entities, part of speech information and valency frames of verbs, nouns and adjectives.

Named Entities: Named Entity recognition denotes the task to learn person names, dates, times, names of organizations, substances or geographical locations. It can be either done to extend existing lexical resources (offline) or to employ this information directly for some NLP application (online) where the first application scenario belongs to the task of knowledge acquisition.

Three major problems have to be solved by the Named Entity recognition:

- recognize which tokens belong to a named entity and which not,
- identify start and end tokens for a named entity since a named entity often consists of several tokens, and
- classify a named entity into a given set of classes (e.g., person, time, location).

Named entity recognition is usually done by employing generative models like Markov chains [Mar13] or discriminative models like Maximum Entropy Modeling [MFP00] and Conditional Random Fields [LMP01]. These models have in common that they represent a text as a graph (chain, in the simple case) of tokens. One major advantage of Conditional Random Fields concerning named entity recognition consists in the fact that they can represent far-distant relationships between named entities, e.g., if two named entities occur several times in a text their assigned named entity type should be identical [SM07].

Part of Speech-Information: Part of speech tagging deals with the problem to assign each token its part-of-speech, e.g., verb, noun, adjective, etc. A part of speech tagger can be used to automatically extend the lexicon or (as a preprocessing step for the parser) to disambiguate words which can be assigned several parts of speech, e.g., the word *can* may be used as a noun or as a verb. Part of speech tagging employs techniques similar as named entity recognition (i.e., Markov Chains, Maximum Entropy Modeling and Conditional Random Fields). One major advantage of Conditional Random Fields is that the label bias problem of the Maximum Entropy Modeling is avoided [LMP01, Wal02]. This problem occurs if the entropy of some states (graph nodes) is very small, e.g., if there exists only a single successor state. In this case their transition probabilities can be almost independent of the output (output here: surface text of a token).

Valency frames: Verbs, nouns and adjectives can have arguments. These arguments can either be obligatory (arguments which must show up) or optional (arguments which need not be present in a sentence). The arguments including their syntactic and semantic features are called the valency frame. An approach to learn the syntactical part of the valency frame by text mining is proposed by Brent [Bre93]. A problem which has to be dealt with is that the usage of parsers (especially of deep parsers) is quite limited for this task since parsers usually require the valency or subcategorization frames already to be known.

4 Learning Readability Formulas

4.1 Readability and Readability Formulas

Various methods to derive a numerical value corresponding to text readability have been proposed [Kla63, DuB04]. One of the most popular readability formulas nowadays, the Flesch Reading Ease score, was developed already in 1948 [Fle48]. For judging readability, this formula uses the average sentence length and word length. The average sentence length is intended to roughly approximate the complexity of a sentence, while the word length is related to word frequency since usually long words are less used. The Flesch readability formula is defined as follows:

$$P = 206.835 - (1.015 \times ASL) - (84.6 \times AWL) \tag{19}$$

with:

P : readability score (scores around zero correspond to simple texts, scores about 100 to difficult texts)

ASL : average sentence length (measured in number of words)

AWL : average word length (measured in syllables)

Later on, this formula was also adapted to German, resulting in the Amstad Readability index [Ams78]. Despite of its age, the Flesch formula is still widely used. Moreover, its indicators sentence/word length are employed in various other readability formulas [Kla63, BV84].

The revised Dale-Chall readability index [CD95] also depends on surface-type indicators. Analogous to Flesch, this index employs (for computing readability) the average sentence length. Instead of recognizing difficult-to-understand words by counting the number of syllables it looks up each word in a certain word list. In the case a word occurs in this list, it is considered to be easy to understand. Thus, the average word complexity is determined by the percentage of words of a text not appearing in this list. To keep this list small, it contains no word forms but only lemmas. Therefore, a lemmatization has to be done before lookup. This allows for instance the word *sleeps* to be found if the list only contains the lemma *sleep*.

The Coh-Metrix-Project is dealing with a certain aspect of text readability of English texts, i.e., the text coherence [MLDM06]. The text coherence is determined by identifying several referential constructs like anaphoras, temporal and spatial relations.

Apart from readability formulas, several readability checkers with graphical user interface were developed which are able to highlight difficult-to-read text passages [CS96, Ras06, vdBH07, HHLO06].

4.2 Machine Learning Techniques

Most of the readability formulas have to be trained with labeled data. For that, a readability study is usually conducted where the readability of texts of a given corpus is determined by a large amount of participants. Besides letting the participants directly rate a text, the following methods are commonly used to derive a readability rating:

- Ask the participants questions about the text. The percentage of correctly answered questions is a good estimation of text readability [ARV71].
- Cloze procedure: Certain words (e.g., every fifth word) are removed from the text. The participants have to fill in the gaps [Tay53].

In the next step, a set of indicators is derived from the texts (e.g., sentence or word length) which describe a certain aspect of readability. The indicators are then used to derive a readability score which as closely as possible matches the ratings obtained from the participants. Many readability formulas calculate the readability score by a linear combination of indicator values (see Section 4.1). In this case, the parameters can be easily determined using linear regression. However, such an approach has the drawback that the influence of each indicator is not obvious and that this procedure can easily lead to overfitting if too many indicators are used. This also requires work for determining a minimal set of indicators.

Therefore, a different approach was followed by the DeLite readability checker [vL07]. Before combining the indicator values, all values are normalized to an interval from zero to one. The parameters of the parameterized sum are all required to be non-negative and sum up to one (thus, they are called weights), i.e., $\sum_{j=1}^{m} w_j = 1$ and $w_j \geq 0$.

In this way, the influence of each indicator can immediately be seen. Furthermore, indicators which are assigned the weight of zero can be automatically removed. In addition, relatively unimportant indicators are guaranteed to have a small influence on the readability formula. Thus, the danger of overfitting is limited.

However, ordinary linear regression can no longer be used to estimate the parameters since standard linear regression does not allow the definition of inequality constraints which are needed to ensure nonnegative weights. DeLite uses the following approach to handle this problem:

- Solve the linear regression with the Lagrange restriction [Gre93] that the weights sum up to one: $\mathbf{L} = (1 \dots 1)^T$ and $q = (1)$ (see Section 2.7).
- Determine all weights which are negative and remove the associated indicators from the regression model.
- Start the regression again.
- Repeat all steps above as long as any negative weights are found.

Besides this iterative linear regression algorithm, DeLite can also follow an alternative approach using robust regression with linear optimization. In contrast to linear regression this approach minimizes the absolute instead of the square error between the readability score and the user ratings [BT97]:

$$\mathbf{w_{opt}} = arg\min_{\mathbf{w}} \sum_{i=1}^{n} |y_i - \mathbf{X_i w}| \tag{20}$$

Additional variables $z_1, \ldots, z_n$ are introduced and the optimization problem is changed in the following way:

$$\begin{aligned} arg\min_{\mathbf{w}} \quad & \sum_{i=1}^{n} z_i && (21)\\ z_i \quad \geq \quad & |y_i - \mathbf{X_i w}| \text{ for } i = 1, \ldots, n && (22) \end{aligned}$$

This problem is equivalent to the original optimization problem since the solutions for the z_i are the lowest numbers which are greater than $|y_i - \mathbf{X_i w}|$. Since

$$z_i \geq |y_i - \mathbf{X_i w}| \Leftrightarrow (z_i \geq y_i - \mathbf{X_i w} \wedge z_i \geq -(y_i - \mathbf{X_i w})) \tag{23}$$

the constraints can be changed to

$$\begin{aligned} z_i \quad & \geq \quad (y_i - \mathbf{X_i w}) && (24)\\ z_i \quad & \geq \quad -(y_i - \mathbf{X_i w}) && (25) \end{aligned}$$

This optimization problem can be solved by common linear optimization algorithms. A commonly used and quite efficient approach for this task is the *Simplex Method*. It reduces the costs continually by traversing the vertices of the polygon which constraints the solution space. Since the cost vector and the constraints are both linear, the solution is guaranteed to be located on such a vertex.

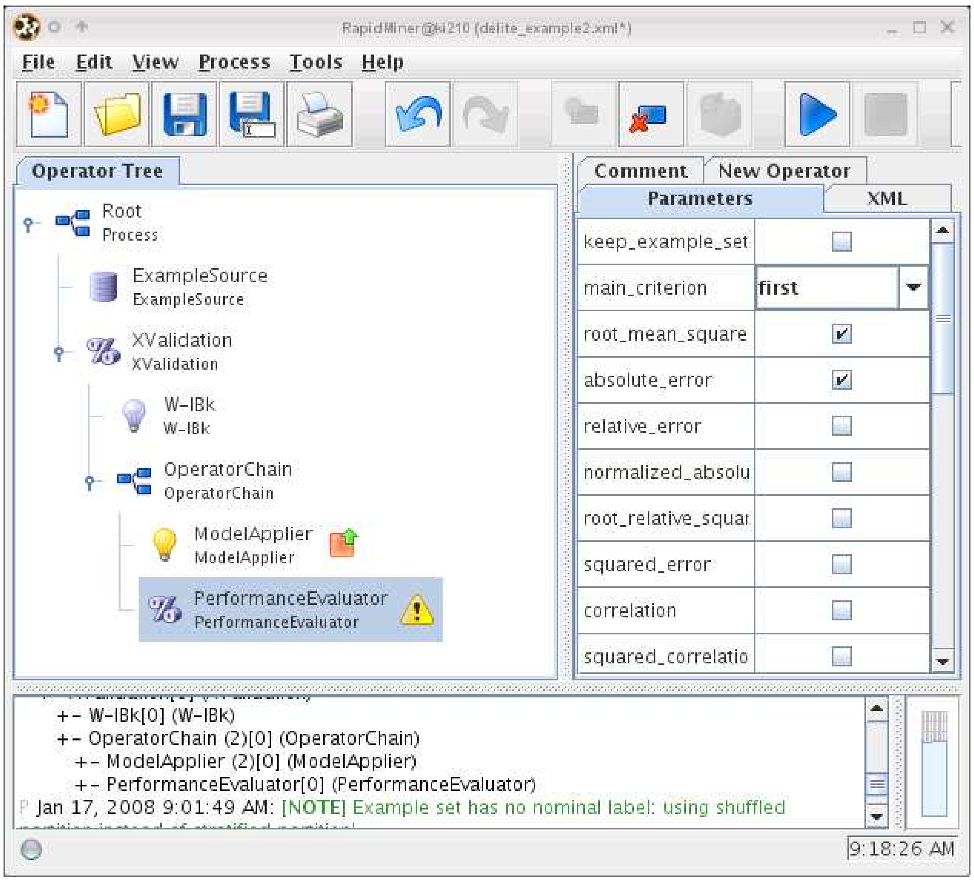
A completely different approach for determining a readability score is proposed by Larsson [Lar06]. He employs support vector machines to separate the vectors of indicator values associated to a given text into the three different readability classes *easy*, *medium* and *difficult*. This method reaches a recall and precision of about 90 %. A drawback of this method is that the classification into three readability levels is rather rough. In contrast, a readability formula based on a parameterized sum as described above provides a continuous readability score.

5 Machine Learning Tools

Two of the most powerful machine learning tools, useful for studying learning algorithms in the fields of knowledge acquisition and readability analysis, are RapidMiner [MWK$^+$06] and Weka [FHH$^+$05]. There exist other tools which support only a few machine learning algorithms. Software of the latter type are for instance the nearest neighbor learner TiMBL (see Section 5.3) and the linear optimization library GLPK (see Section 5.4). The most important features of all tools mentioned above are compared in Table 5.

Table 5: Comparison of the functionality of several machine learning tools.

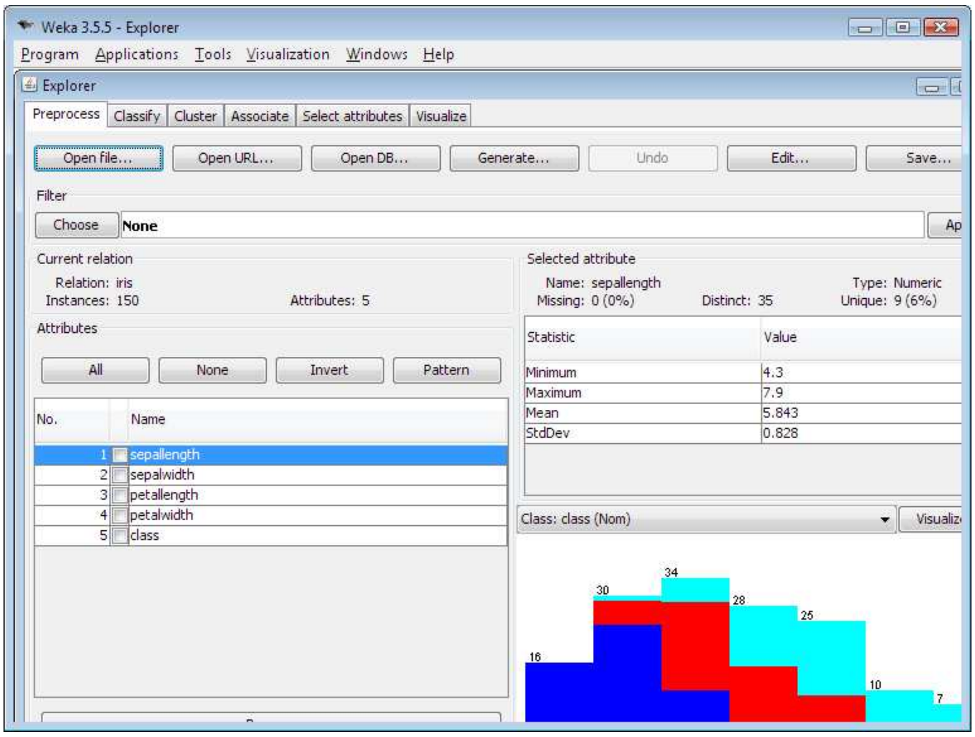
Feature/Tools	RapidMiner	Weka	TiMBL	GLPK	FOIL
Nearest neighbor	✓	✓	✓	-	-
Support vector machines	✓	✓	-	-	-
Clustering	✓	✓	-	-	-
Decision tree	✓	✓	✓	-	-
Neural network	✓	✓	-	-	-
Linear regression	✓	✓	-	-	-
Naïve bayes classifier	✓	✓	-	-	-
Conditional random fields	✓	-	-	-	-
Linear optimization	✓	-	-	✓	-
Relational rule learning	-	-	-	-	✓
Programming language	Java	Java	C	C	C
GUI	Yes	Yes	No	No	No



keep_example_set	☐
main_criterion	first
root_mean_square	✓
absolute_error	✓
relative_error	☐
normalized_absolu	☐
root_relative_squar	☐
squared_error	☐
correlation	☐
squared_correlatio	☐

```
  +- W-IBk[0] (W-IBk)
  +- OperatorChain (2)[0] (OperatorChain)
    +- ModelApplier (2)[0] (ModelApplier)
    +- PerformanceEvaluator[0] (PerformanceEvaluator)
P Jan 17, 2008 9:01:49 AM: [NOTE] Example set has no nominal label: using shuffled
```

Figure 11: Screenshot of the machine learning workbench RapidMiner.


Weka 3.5.5 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose
Ap
All None Invert Pattern
Visualiz
16
30
34
28
25
10
7

None

Current relation
Relation: iris
Instances: 150 Attributes: 5

Selected attribute
Name: sepallength Type: Numeric
Missing: 0 (0%) Distinct: 35 Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Class: class (Nom)

Figure 12: Screenshot of the machine learning workbench Weka.

5.1 RapidMiner

RapidMiner (formerly called Yale) was originally developed by the Department of Artificial Intelligence of the University Dortmund. Currently, the development is continued by a spin-off company of this university. RapidMiner[4] contains more than 400 operators for data mining and machine learning. Moreover, it also provides all operators of Weka.

RapidMiner includes a graphical user interface which offers a wide range of settings (see Figure 11) and allows to comfortably combine and validate different machine learning operators. In the example shown in Figure 11, a cross-validation operator (XValidation) is specified which evaluates MAE (mean absolute error) and RMSE (root mean square error) [GH62] for a calculation with a nearest neighbor algorithm from Weka (W-IBK). Before the operator *ModelApplier* a breakpoint is set which is symbolized by a red rectangle.

5.2 Weka

Weka[5] is a machine learning workbench which provides a functionality similar as RapidMiner. It supports most of the RapidMiner machine learning algorithms and provides a graphical user interface as well.

[4]URL: `http://www.rapidminer.com`

[5]URL: `http://www.cs.waikato.ac.nz/ml/weka/`

5.3 TiMBL

5.3.1 Overview

TiMBL[6] was developed at the University of Tilburg and focuses on nearest neighbor classification [DZvdSvdB07]. It can be accessed via a server interface which allows for an easy integration into other applications. In contrast to RapidMiner and Weka, TiMBL provides no GUI.

5.3.2 Classification Process

The classification of a previously unseen vector is basically done in two steps. First the k-nearest neighbors of this vector are determined using a given similarity measure. The class is then chosen by a majority vote of the class labels of these neighbors. TiMBL employs the 1-norm of the vector difference as similarity measure where all vector components are first normalized to the interval $[0, 1]$ by a linear transformation. The difference of two non-numerical vector components is defined to be 0 if both components are identical, to be 1 otherwise. For character strings the Levenshtein distance [Lev65] between both strings can be alternatively used as a difference measure. Note that the individual vector components are automatically weighted according to their importance where several weighting algorithms are provided for this purpose (e.g., information-gain, information-ratio, or chi-squared).

5.4 GLPK

GLPK (GNU Linear Programming Kit) is a library for linear optimization and can solve continuous-valued, integer-valued and boolean-valued optimization problems. It provides both the Simplex Algorithm and interior point methods [BT97]. The latter methods have the advantage that a solution is guaranteed to be found in polynomial time which is not the case for the Simplex Algorithm. However, for most practical problems the Simplex Algorithm is still very efficient. Since GLPK has a high performance, it is capable of solving large optimization problems.

5.5 FOIL

FOIL is the abbreviation for ***F****irst* ***O****rder* ***I****nductive* ***L****earner* and denotes a system which is able to learn horn clauses from a given collection of training examples in form of instantiated predicates [Qui90]. FOIL is a relational learning systems (see Section 2.9) which means that the literals learned by FOIL are usually not atomic but contain variables as arguments which can be bound to any constants. Furthermore, certain constants can be explicitely allowed to occur in these literals as well. FOIL is also able to find recursive rules where the predicate of the conclusion occurs additionally in a literal of the premise.

6 Conclusions

Several approaches for the application of machine learning algorithms for knowledge acquisition and readability analysis were introduced.

The comparison of the knowledge acquisition methods showed that almost all of these approaches are based on a surface representation or on a syntactic dependency

[6]TiMBL is the abbreviation for **Ti**lburg **M**emory **B**ased-**L**earner.

tree. In contrast to that, approaches based on semantic representations are very sparse. We expect that an algorithm based on a deep semantic representation can improve recall (essentially the number of semantic relations/entailments learned from the text) and precision (essentially the number of semantic relations/entailments correctly learned from the text) considerably.

Currently used readability formulas are mainly based on a simple linear combination of several readability indicators. The parameters are typically learned employing a linear regression. However, such an approach has serious drawbacks concerning overfitting or an easy interpretation of parameters. Also, a weighted sum is quite limited in its capabilities to approximate a given target function. Thus, non-linear readability formulas are expected to gain importance in the future. The DeLite readability formula which is developed by the IICS also employs non-linear optimization techniques.

Finally, several existing and freely available machine learning tools, which are suitable for knowledge acquisition and learning of readability formulas, were compared with each other. The largest functionality is provided by Weka and RapidMiner. Both tools support all of the most popular machine learning algorithms like regression, neural networks, decision trees, support vector machines etc. Furthermore, they also provide an easy-to-use graphical user interface.

Acknowledgments

We especially are obliged to Ingo Glöckner, Sven Hartrumpf, Hermann Helbig, and Rainer Osswald for proof-reading this document. Furthermore, we also thank all other colleagues in our group who contributed to this work as well, including Christoph Doppelbauer, Christian Eichhorn, and Johannes Leveling.

References

[Ada06] Rod Adams. Textual Entailment through extended lexical overlap. In *Proceedings of the 2nd ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 128–133, Venice, Italia, 2006.

[AIMO07] Eiji Aramaki, Takeshi Imai, Kengo Miyo, and Kazuhiko Ohe. SVM–based semantic relation classification using physical sizes. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval)*, pages 464–467, Prague, Czech Republic, 2007.

[Ams78] Tony Amstad. *Wie verständlich sind unsere Zeitungen?* PhD thesis, Universität Zürich, 1978.

[ARV71] H. Anger, R.Bargmann, and M. Voigt. *Verständiges Lesen VL 7-9*. Julius Beltz, Weinheim, Germany, 1971.

[BBQ03] Christian Biemann, Stefan Bordag, and Uwe Quasthoff. Lernen paradigmatischer Relationen auf iterierten Kollokationen. In *Proceedings of the GermaNet-Workshop 2003*, pages 87–94, Tübingen, Germany, 2003.

[BBQ04] Chris Biemann, Stefan Bordag, and Uwe Quasthoff. Automatic acquisition of paradigmatic relations using iterated co-occurrences. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Lisbon, Portugal, 2004.

[BDK$^+$03] Stephan Busemann, Witold Drozdynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, Hans Uszkoreit, Feiyu Xu, Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. Integrating information extraction and automatic hyperlinking. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 2*, pages 117 – 120, Sapporo, Japan, 2003.

[BL03] Regina Barzilay and Lillian Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 16–23, 2003.

[BM01] Regina Barzilay and Kathleen R. McKeown. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 50–57, Toulouse, France, 2001.

[BM06] Johann Bos and Katja Markert. Recognizing textual entailment with robust logical inference. In Joaquin Quinonero-Candela et al., editor, *Machine Learning Challenges*, number 3944 in LNAI, pages 404–426. Springer, Berlin, Germany, 2006.

[Bre93] Michael Brent. From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19(3):243–262, 1993.

[BT97] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts, USA, 1997.

[BV84] Richard Bamberger and Erich Vanecek. *Lesen - Verstehen - Lernen - Schreiben.* Diesterweg, Frankfurt am Main, Germany, 1984.

[CD95] Jeanne Chall and Edgar Dale. *Readability Revisited: The New Dale-Chall Readability Formula.* Brookline Books, Brookline, Massachusetts, USA, 1995.

[CF98] George Miller Christiane Fellbaum. *WordNet. An electronic lexical database.* MIT Press, Cambridge, Massachusetts, USA, 1998.

[CGH$^+$94] Kenneth Church, William Gale, Patrick Hanks, Donal Hindle, and Rosamund Mood. Lexical substitutability. In Beth T.S. Atkins and Antonio Zampolli, editors, *Computational Approaches to the Lexicon*, pages 153–177. Oxford University Press, 1994.

[Cla98] German Rigau Claramunt. *Automatic Acquisition of Lexical Knowledge from Machine Readable Dictionaries.* PhD thesis, Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 1998.

[Cos07] Fintan J. Costello. UCD-FC: Deducing semantic relations using WordNet senses that occur frequently in a database of noun-noun compounds. In *SemEval 2007: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 370–373, Prague, Czech Republic, 2007.

[CPSTS05] Philip Cimiano, Aleksander Pivk, Lars Schmidt-Thieme, and Steffen Staab. Learning taxonomic relations from heterogeneous sources of evidence. In Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini, editors, *Ontology Learning from text: Methods, evaluation and applications*, pages 59–73. IOS Press, Amsterdam, The Netherlands, 2005.

[CS96] Raman Chandrasekar and Bangalore Srinivas. Automatic induction of rules for text simplification. Technical Report IRCS Report 96-30, University of Pennsylvania, Philadelphia, Pennsylvania, USA, 1996.

[DLP97] Ido Dagan, Lillian Lee, and Fernando Pereira. Similarity–based methods for word sense disambiguation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 56–63, Madrid, Spain, 1997.

[DuB04] William H. DuBay. Principles of readability. Unpublished, online available at `http://www.impact-information.com/impactinfo/readability02.pdf`, 2004.

[Dun93] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1), 1993.

[Dže07] Sašo Džeroski. Inductive logic programming in a nutshell. In Lise Getoor and Ben Taskar, editors, *Statistical Relational Learning*, pages 57–92. MIT Press, Cambridge, Massachusetts, USA, 2007.

[DZvdSvdB07] Walter Daelemans, Jakub Zavrel, Koo van der Sloot, and Antal van den Bosch. TiMBL: Tilburg memory based learner, version 6.1, reference guide. Technical Report 07-07, University Tilburg, ILK Research Group, Tilburg, The Netherlands, 2007.

[Edm97] Philip Edmonds. Choosing the word most typical in context using a lexical co-occurrence network. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics (ACL)*, pages 507–509, Madrid, Spain, 1997.

[Fel98] Christiane Fellbaum. *WordNet- An Electronic Lexical Database*. MIT Press, Cambridge, Massachusetts, USA, 1998.

[FHH$^+$05] Eibe Frank, Mark A. Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H. Witten, and Leonhard Trigg. WEKA - a machine learning workbench for data mining. In Oded Maimon and Lior Rokach, editors, *The Data Mining and Knowledge Discovery Handbook*, pages 1305–1314. Springer, Berlin, Germany, 2005.

[Fle48] Rudolf Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32:221–233, 1948.

[FRH02] Victoria Fromkin, Robert Rodman, and Nina Hyams. *Introduction to Language*. Heinle, Frankfurt (Main), Germany, 2002.

[GBM06] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135, 2006.

[GH62] J. Arthur Greenwood and H.O. Hartley. *Guide to Tables in Mathematical Statistics*. Princeton University Press, Princeton, New Jersey, USA, 1962.

[GHH06] Ingo Glöckner, Sven Hartrumpf, and Hermann Helbig. Automatic knowledge acquisition by semantic analysis and assimilation of textual information. In *Proceedings of KONVENS 2006*, pages 36–43, Konstanz, Germany, 2006.

[GHO05] Ingo Glöckner, Sven Hartrumpf, and Rainer Osswald. From GermaNet glosses to formal meaning postulates. In B. Fisseni, H.-C. Schmitz, B. Schröder, and P. Wagner, editors, *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen – Beiträge zur GLDV-Tagung 2005 in Bonn*, pages 394–407. Peter Lang, Frankfurt (Main), Germany, 2005.

[Gli06] Alfio Massimiliano Gliozzo. The GOD model. In *Poster Proceedings of the 11th Conference on the European Chapter of the Association for Computational Linguistics (EACL)*, pages 147–158, Trento, Italy, 2006.

[GM98] Moises Goldszmidt and Sahami Mehran. A probabilistic approach to full-text document clustering. Technical Report ITAD-433-MS-98-044, SRI International, Stanford University, 1998.

[GMDD07] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the 3rd ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 177–190, Santa Fe, USA, 2007.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Pennsylvania, USA, 1989.

[Gre93] William Greene. *Econometric Analysis*. Prentice Hall, Englewood Cliffs, New York, USA, 1993.

[Har68] Zelling Harris. *Mathematical Structures of Language*. John Wiley & Sons, New York, USA, 1968.

[Har03] Sven Hartrumpf. *Hybrid Disambiguation in Natural Language Analysis*. PhD thesis, FernUniversität in Hagen, Osnabrück, Germany, 2003.

[Hea92] Marti Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 539–545, Nantes, France, 1992.

[Hel06] Hermann Helbig. *Knowledge Representation and the Semantics of Natural Language*. Springer, Berlin, Germany, 2006.

[HHLO06] Sven Hartrumpf, Hermann Helbig, Johannes Leveling, and Rainer Osswald. An architecture for controlling simple language in web pages. *eMinds: International Journal on Human-Computer Interaction*, 1(2):93–112, 2006.

[Hin90] Donald Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 268–275, Pittsburgh, Pennsylvania, USA, 1990.

[HLQW01] Gerhard Heyer, Martin Läuter, Uwe Quasthoff, and Christian Wolff. Wissensextraktion durch linguistisches Preprocessing bei der Korpusanalyse. In *Proceedings der GLDV-Frühjahrstagung*, pages 71–83, Gießen, Germany, 2001.

[IKN06] Diana Inkpen, Darren Kipp, and Vivi Nastase. Machine learning experiments for textual entailment. In *Proceedings of the 2nd ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 10–15, Venice, Italia, 2006.

[KD96] Alistair Knott and Robert Dale. Choosing a set of coherence relations for text generation: A data-driven approach. In *Trends in Natural Language Generation An Artificial Intelligence Perspective*, number 1036 in LNCS. Springer, Berlin, Germany, 1996.

[KL51] Solomon Kullback and Richard Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[Kla63] George R. Klare. *The Measurement of Readability.* Iowa State University Press, Ames, Iowa, USA, 1963.

[KM06] Zornitsa Kozareva and André Montoyo. Mlent: The machine learning entailment system of the university of alicante. In *Proceedings of the 2nd ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 16–21, Venice, Italia, 2006.

[KRTS05] Vipul Kashyap, Cartic Ramakrishnan, Christopher Thomas, and Amit P. Sheth. Texaminer: An experimentation framework for automated taxonomy bootstrapping. *International Journal of Web and Grid Services*, 1(2), 2005.

[Lar06] Patrik Larsson. Classification into readability levels. Master's thesis, Department of Linguistics and Philology, University Uppsala, Uppsala, Sweden, 2006.

[Lee99] Lillian Lee. Measures of distributional similarity. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, College Park, Maryland, USA, 1999.

[Lev65] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965. (Russian). English translation in: Soviet Physics Doklady, 10(8) S. 707-710, 1966.

[LFL98] Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.

[LMP01] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, Pittsburgh, Pennsylvania, USA, 2001.

[LP01] Dekang Lin and Patrick Pantel. Dirt - discovery of inference rules from text. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 323–328, San Francisco, California, USA, 2001.

[LPJS04] Cupertino Lucero, David Pinto, and Héctor Jiménez-Salazar. An automatic method to identify antonymy relations. In *Proceedings of the Workshop on Lexical Resources and the Web for Word Sense Disambiguation of the 9th Ibero-American Conference on Artificial Intelligence (IBERAMIA)*, pages 105–111, Puebla, Mexico, 2004.

[LZHST02] Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, and John Shawe-Taylor. The perceptron algorithm with uneven margins. In *Proceedings of the International Conference on Machine Learning*, pages 379–386, 2002.

[Mac05] David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, Cambridge, UK, 2005.

[Mar13] Andrei A. Markov. An example of statistical investigation in the text of 'eugene onyegin' illustrating coupling of 'tests' in chain. In *Proceedings of the Academy of Sciences*, pages 153–162, St. Petersburg, Russia, 1913.

[Mas94] Toshiyuki Masui. Evolutionary learning of graph layout constraints from examples. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 103–108, Marina del Rey, California, USA, 1994.

[MFP00] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the 7th International Conference on Machine Learning*, pages 591–598, Stanford, California, USA, 2000.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Companies, New York, New York, USA, 1997.

[MJ04] Emmanuel Morin and Christian Jaquemin. Automatic acquisition and expansion of hypernym links. *Computers and the Humanities*, 38(4):363–396, 2004.

[MLDM06] Philip M. McCarthy, Erin J. Lightman, David F. Dufty, and Danielle S. McNamara. Using coh-metrix to assess distributions of cohesion and difficulty: An investigation of the structure of high-school textbooks. In *Proceedings of the 28th annual conference of the Cognitive Science Society*, Vancouver, Canada, 2006.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, USA, 1999.

[MWK$^+$06] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, pages 935–940, 2006.

[PP06] Patrick Pantel and Marco Pennachiottti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING) and 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sydney, Australia, 2006.

[QHFC04] Thanh Tho Quan, Siu Cheung Hui, A.C.M. Fong, and True Hoang Cao. Automatic generation of ontology for scholarly semantic web. In *The Semantic Web - ISWC 2004*, volume 4061 of *LNCS*, pages 726–740. Springer, Berlin, Germany, 2004.

[Qui90] Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Ras06] Ecaterina Rascu. A controlled language approach to text optimization in technical documentation. In *Proceedings of KONVENS 2006*, pages 107–114, Konstanz, Germany, 2006.

[RDH$^+$83] Fabio Rinaldi, James Dowdall, Michael Hess, Diego Mollá, Rolf Schwitter, and Kaarel Kaljurand. Knowledge-based question answering. In *Proceedings of the first conference on Applied natural language processing*, pages 73–80, Santa Monica, California, USA, 1983.

[RH02] Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 41–47, Philadelphia, Pennsylvania, USA, 2002.

[Rug97] Gerda Ruge. Automatic detection of thesaurus relations for information retrieval applications. In *Lecture Notes in Computer Science 1337*. Springer, Berlin, Germany, 1997.

[Rya97] Thomas P. Ryan. *Modern Regression Methods*. John Wiley & Sons, New York, USA, 1997.

[SBO07] Richard Socher, Chris Biemann, and Rainer Osswald. Combining contexts in lexicon learning for semantic parsing. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, pages 175–182, Copenhagen, Denmark, 2007.

[SD07] Idan Szpektor and Ido Dagan. Learning canonical forms of entailment rules. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 151–170, 2007.

[SDC04] Idan Szpektor, Ido Dagan, and Bonaventura Coppola. Scaling web-based acquisition of entailment relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic, 2004.

[Sek05] Satoshi Sekine. Automatic paraphrase discovery based on context and keywords between NE pairs. In *Proceedings of the International Workshop on Paraphrase (IWP)*, Jeju Island, Korea, 2005.

[SHK06] Sangweon Suh, Harry Halpin, and Edan Klein. Extracting common sense knowledge from wikipedia. In *Proceedings of the Web Content Mining with Human Language Technologies Conference*, Athens, Georgia, USA, 2006.

[SJN05] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, Cambridge, Massachusetts, USA, 2005.

[SM07] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Statistical Relational Learning*, pages 93–127. MIT Press, Cambridge, Massachusetts, USA, 2007.

[SSG03] Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 224–231, Sapporo, Japan, 2003.

[Tay53] W. Taylor. Cloze procedure: A new tool for measuring readability. *Journalism Quarterly*, 30:415–433, 1953.

[Vap79] Vladimir Vapnik. *Estimation of Dependencies Based on Empirical Data [in Russian]*. Nauka, Moscow, Russia, 1979. English translation: Springer, Berlin, 1982.

[vdBH07] Tim vor der Brück and Sven Hartrumpf. A semantically oriented readability checker for German. In *Proceedings of the Language and Technology Conference (L&TC)*, pages 270–274, Poznań, Poland, 2007.

[vL07] Tim vor der Brück and Johannes Leveling. Parameter learning for a readability checking tool. In *Proceedings of the LWA 2007 (Lernen-Wissen-Adaption), Workshop KDML*, pages 149–153. Gesellschaft für Informatik, Halle (Saale), Germany, 2007.

[Wal02] Hanna Wallach. Efficient training of conditional random fields. Master's thesis, School of Cognitive Science, University of Edinburgh, 2002.

[WCH87] Morten Winston, Roger Chaffin, and Douglas Hermann. A taxonomy of part-whole relations. *Cognitive Science*, 11(4):417–444, 1987.

[WWM04] Julie Weeds, David Weir, and Diana McCarthy. Characterizing measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, Geneva, Switzerland, 2004.

[ZM06] Fabio Massimo Zanzotto and Alessandro Moschitti. Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 401–408, Sydney, Australia, 2006.

[ZZL07] Shiqi Zhao, Ming Zhou, and Ting Liu. Learning question paraphrases for QA from encarta logs. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1795–1800, Hyderabad, India, 2007.

List of Figures

List of Tables