Preface

Here come the golden words.

Place, Month Year *First Name, Surname*

Table of Contents

List of Contributors

Alexander Maedche
FZI
University of Karlsruhe
Germany
http://www.fzi.de/wim

Steffen Staab
Institute AIFB
University of Karlsruhe
Germany
http://www.aifb.uni-karlsruhe.de/WBS
and
Learning Lab Lower Saxony
Hannover, Germany
http://www.learninglab.de
and
Ontoprise GmbH
Karlsruhe, Germany
http://www.ontoprise.de

Viktor Pekar
Bashkir State University
Okt.Revolutsii 3a
Ufa, Russia, 450000
vpekar@ufanet.ru

1. Ontology Learning Part One — On Discovering Taxonomic Relations from the Web

Alexander Maedche[1], Viktor Pekar[2], and Steffen Staab[3]

[1] FZI
University of Karlsruhe
Germany

[2] Bashkir State University
Okt.Revolutsii 3a
Ufa, Russia, 450000

[3] Institute AIFB
University of
Karlsruhe
Germany

&

Learning Lab
Lower Saxony
Hannover,
Germany

&

Ontoprise GmbH
Karlsruhe, Germany

Summary.

Ontologies may help to facilitate the finding and use of Web information. However, the engineering of an ontology may turn out expensive and time-consuming. Therefore, we exploit ontology learning techniques that automate ontology engineering to some extent. In this chapter, we focus on the learning of the taxonomic backbone of ontologies, presenting a survey on algorithms as well as some new ideas that consider the structure of existing ontology parts. Eventually, we describe an evaluation of our proposal and give concrete results.

1.1 Introduction

The Web in its current form is an impressive success with a growing number of users and information sources. However, the growing complexity of the Web is not reflected in the current state of Web technology. The heavy burden of accessing, extracting, interpretating and maintaining information is left to the human user. Tim Berners-Lee, the inventor of the WWW, coined the vision of a Semantic Web in which background knowledge on the meaning of Web resources is stored through the use of machine-processable (meta-)data. The Semantic Web should bring structure to the content of Web pages, being an extension of the current Web, in which information is given a well-defined meaning. Thus, the Semantic Web will be able to support automated services based on these descriptions of semantics. These descriptions are seen as a key factor to finding a way out of the growing problems of traversing the expanding Web space, where most Web resources can currently only be found through syntactic matches (e.g., keyword search), providing a new level of Web Intelligence.

Ontologies have shown to be the right answer to these problems by providing a formal conceptualization of a particular domain that is shared by a group of people. Thus, in the context of the Semantic Web, ontologies describe domain theories for the explicit representation of the semantics of the data. The Semantic Web relies

heavily on these formal ontologies that structure underlying data enabling comprehensive and transportable machine understanding. Though ontology engineering tools have matured over the last decade, the manual building of ontologies still remains a tedious, cumbersome task which can easily result in a knowledge acquisition bottleneck. The success of the Semantic Web strongly depends on the proliferation of ontologies, which requires that the engineering of ontologies is completed quickly and easily. When using ontologies as a basis for Semantic Web applications, one has to face exactly this issue and in particular questions about development time, difficulty, confidence and the maintenance of ontologies. Thus, what one ends up with is similar to what knowledge engineers have dealt with over the last two decades when elaborating methodologies for knowledge acquisition or workbenches for defining knowledge bases [1.24, 1.25]. A method which has proven to be extremely beneficial for the knowledge acquisition task is the integration of knowledge acquisition with machine learning techniques

In this chapter we focus on an essential part of ontology engineering, namely the development of the taxonomic backbone of the ontology. The purpose of the chapter is to give a survey of existing work on learning taxonomic relations from texts and an example of how such learning may be performed and evaluated.

The article is organized as following. We start with two survey on existing work (also cf. [1.17]). We consider symbolic (Section 1.2) and statistics-based approaches (Section 1.3). The trade-off between the two is that statistics-based approaches allow for better scaling, but symbolic approaches might eventually turn up being more precise. Therefore, we aim at a reconciliation between the two paradigms and propose new algorithms for taxonomy learning including existing taxonomic relations as background knowledge (Section 1.4). The Sections 1.5 through 1.7 will focus on the evaluation of algorithms for ontology learning and elucidate the evaluation with a typical Web scenario. In particular, Section 1.5 introduces the overall setting that we used as an example for evaluating the learning of taxonomies, presenting the input data for our learning method. Section 1.6 shows how to perform evaluation and finally Section 1.7 gives the results we have obtained so far.

1.2 Survey of Symbolic Approaches

1.2.1 Extraction of Taxonomic Relations

The idea of using lexico-syntactic patterns in the form of regular expressions for the extraction of semantic relations, in particular taxonomic relations has been introduced by Hearst [1.7]. Pattern-based approaches in general are heuristic methods using regular expressions that originally have been successfully applied in the area of information extraction (see [1.9]). In this lexico-syntactic ontology learning approach the text is scanned for instances of distinguished lexico-syntactic patterns that indicate a relation of interest, e.g. the taxonomic relation. Thus, the underlying idea is very simple: Define a regular expression that captures re-occurring expres-

sions and map the results of the matching expression to a semantic structure, such as taxonomic relations between concepts.

Example. This examples provides a sample pattern-based ontology extraction scenario. For example, in [1.7] the following lexico-syntactic pattern is considered

$$\ldots NP\{, NP\} * \{, \} \text{ or other } NP \ldots$$

When we apply this pattern to a sentence it can be inferered that the NP's referring to concepts on the left of *or other* are sub concepts of the NP referring to a concept on the right. For example from the sentence

Bruises, wounds, broken bones or other injuries are common.

we extract the taxonomic relations (BRUISE,INJURY), (WOUND,INJURY), and, (BROKEN–BONE,INJURY).

In [1.7] the patterns have been defined manually, which is a time-consuming and error-prone task. In [1.19] the work proposed by [1.7] is extended by using a symbolic machine learning tool to refine lexico-syntactic patterns. In this context the PROMETHEE system has been presented that supports the semi-automatic acquisition of semantic relations and the refinement of lexico-syntactic patterns.

Assadi's work [1.1] reports a practical experiment of construction of a regional ontology in the field of electric network planning. He describes a clustering approach that combines linguistic and conceptual criteria. As an example he gives the pattern <NP, line> which results in two categorizations by modifiers. The first categorization is motivated by the `function_of_structure` modifiers, resulting in a clustering of `connection line`, `dispatching line` and `transport line` (see Table 1.1). For the other concepts the background knowledge lacks adequate specifications such that further categorizations could have been proposed.

Table 1.1. Example Categorization

A proposal categorization	The other candidate terms
connection line	mountain line
dispatching line	telecommunication line
transport line	input line

Faure and Nedellec [1.3] have presented a cooperative machine learning system called ASIUM which is able to acquire taxonomic relations from syntactic parsing. The ASIUM system is based on a conceptual clustering algorithm. Basic clusters are formed on head words that occur with the same verb after the same preposition. ASIUM successively aggregates clusters to form new concepts and the hierarchies of concepts form the ontology.

An ontology learning system where the different techniques have been applied to dictionary definitions in the context of the insurance and telecommunication domains is described in [1.12, 1.16]. An important aspect in this system and approach

is that existing concepts are included in the overall process. Thus, in contrast to [1.7, 1.19] the extraction operations have been performed on the concept level, thus, patterns have been directly matched onto concepts. Thus, the system is, besides extracting taxonomic relations from scratch, able to refine existing relations and refer to existing concepts.

1.2.2 Refinement of Taxonomic Relations

Hahn and Schnattinger [1.5] introduced a methodology for the maintenance and refinement of domain-specific taxonomies. An ontology is incrementally updated as new concepts are acquired from real-world texts. The acquisition process is centered around linguistic and conceptual "quality" of various forms of evidence underlying the generation and refinement of concept hypotheses. In particular they consider semantic conflicts and analogous semantic structures from the knowledge base into the ontology in order to determine the quality of a particular proposal. Thus, they extend an existing ontology with concepts and taxonomic relations between concepts.

The system Camille[1] was developed as a natural language understanding system, e.g. when the parser comes across words that it does not know, Camille tries to infer whatever it can about the meaning of the unknown word [1.6]. If the unknown word is a noun, semantic constraints on slot-fillers provided by verbs give useful limitations about what the noun could mean. The meaning of a noun can be derived, because constraints are associated with verbs. Learning unknown verbs is more difficult, thus, verb acquisition has been the main focus of the research on Camille. Camille was tested on several real-world domains within information extraction tasks (MUC), where the well-known scoring methods precision and recall, taken from the information retrieval community, have been calculated. For the lexical acquisition task *recall* is defined as the precentage of correct hypobook. A hypobook was counted as correct if one of the concepts in the hypobook matched the target concept. Precision is the total number of correct concepts divided by the number of concepts generated in all the hypobook. Camille has achieved a recall of 42% and a precision of 19% on a set of 50 randomly-selected sentences containing 17 different verbs.

1.3 Survey of Statistics-based Approaches

In this section we will consider those approaches to taxonomy learning that infer the semantics of a new concept and relate it to concepts already present in the ontology on the basis of statistical data about cooccurrence behavior of word(s) expressing these concepts. The main idea common to these approaches is that the semantic identity of a word is reflected in its distribution over different contexts, so that the meaning of a word is represented in terms of words cooccurring with it and the frequencies of the cooccurrences. This manner of representing semantics obviates

[1] Contextual Acquisition Mechanism for Incremental Lexeme Learning

the need to prepare special resources to process it, such as lexico-syntactic patterns, and promises to make the process of taxonomy learning fully automatic. However, the main problem for these approaches is the notorious data sparseness, i.e. the fact corpus data available on words of interest may not be indicative of their meaning. Correspondingly, the major efforts here are spent on the development of ways to sort and effectively use available cooccurrence data.

1.3.1 Statistics-based Extraction of Taxonomic Relations

Distributional data about words may be used to build concepts and their embedding into a hierarchy "from scratch". In this case a certain clustering technique is applied to distributional representations of concepts. Clustering can be defined as the process of organizing objects into groups whose members are similar in some way (see [1.11]). In general there are two major styles of clustering: **non-hierarchical clustering** in which every object is assigned to exactly one group and **hierarchical clustering**, in which each group of size greater than one is in turn composed of smaller groups. Hierarchical clustering algorithms are preferable for detailed data analysis. They produce hierarchies of clusters, and therefore contain more information than non-hierarchical algorithms. However, they are less efficient with respect to time and space than non-hierarchical clustering[2]. [1.18] identify two main uses for clustering in natural language processing[3]: The first is the use of clustering for exploratory data analysis, the second is for generalization. Seminal work in this area of so-called *distributional clustering* of English words has been described in [1.20]. Their work focuses on constructing class-based word co-occurrence models with substantial predictive power. In the following the existing and seminal work of applying statistical hierarchical clustering in NLP (see [1.20]) is adopted and embedded into the framework.

Baseline Hierarchical Clustering. The tree of hierarchical clusters can be produced either bottom-up, by starting with individual objects and grouping the most similar ones, or top-down, whereby one starts with all the objects and divides them into groups.

Algorithm 1 given in the following (adopted from [1.18]) describes the bottom-up algorithm. It starts with a separate cluster for each object. In each step, the two most similar clusters are are determined, and merged into a new cluster. The algorithm terminates when one large cluster containing all objects has been formed. The most important aspect in clustering is the selection of an appropriate computation strategy and a similarity measure. We will introduce a number of computation strategies (e.g. single-link, complete link or group-average) and similarity measures (e.g. cosine, Kullback Leibler) later in this subsection.

Algorithm 2 given in the following (adopted from [1.18]) roughly describes the top-down algorithm. It starts out with one cluster that contains all objects. The algorithm then selects the least coherent cluster in each iteration and splits it. Clusters

[2] Hierarchical clustering has in the average quadratic time and space complexity.

[3] A comprehensive survey on applying clustering in NLP is also available in the EAGLES report, see http://www.ilc.pi.cnr.it/EAGLES96/rep2/node37.htm

Algorithm 1 Hierarchical Clustering Algorithm — Bottom-Up

Require: a set $X = \{x_1, \ldots, x_n\}$ of objects, n as the overall number of objects,
a function sim: $2^X \times 2^X \rightarrow R$
Ensure: the set of clusters K (or cluster hypothesis)

for i:=1 to n **do**
$k_i := x_i$.
end for
$K := \{k_1, \ldots, k_n\}$.
$j := n + 1$.
while $|K| > 1$ **do**
(k_{n1}, k_{n2}) :=arg $\max_{(k_u,k_v) \in K \times K}$ sim(k_u, k_v).
$k_j = k_{n1} \cup k_{n2}$.
$K := K \backslash \{k_{n1}, k_{n2}\} \cup \{k_j\}$.
$j := j + 1$
end while

with similar objects are more coherent than clusters with dissimilar objects. Thus, the strategies single-link, complete link and group-average can also serve as measures of cluster coherence (function coh) in top-down clustering.

Algorithm 2 Hierarchical Clustering Algorithm — Top-Down

Require: a set $X = \{x_1, \ldots, x_n\}$ of objects, n as the overall number of objects,
a function coh: $2^X \longrightarrow R$
a function split: $2^X \times 2^X \rightarrow 2^X$
$K := \{X\} (= k_1)$
$j := 1$
while $\exists k_i \in K s.t. |k_i| > 1$ **do**
$k_u :=$ arg $\min_{k_v \in K}$coh(k_v)
(k_{j+1}, k_{j+2}) = split(k_u)
$K := K \setminus \{k_u\} \cup \{k_{j+1}, k_{j+2}\}$
$j := j + 2$.
end while

The reader may note that splitting a cluster (function split) is also a clustering task (namely the task of finding two sub-clusters of a cluster). Thus, there is a recursive need for a second clustering algorithm. Any clustering algorithm may be used for the splitting operation, including bottom-up algorithms.

As mentioned earlier an important aspect is the selection of an appropriate computation strategy and a similarity measure. In the following the most important ones are presented.

Computation strategies used in hierarchical clustering. In this work it is focused on the three functions single link, complete link and group-average that have shown to perform good in statistical hierarchical clustering. Their advantages and disadvantages a shortly introduced. The interested reader is referred to a more detailed introduction given in [1.11]. Measuring similarity based on **single linkage** means

that the similarity between two clusters is the similarity of the two closest objects in the clusters. Thus, one has to search over all pairs of objects that are from the two different clusters and select the pair with the greatest similarity. Single-link clustering have clusters with local coherence. If similarity is based on **complete linkage** the similarity between two clusters is computed based on the similarity of the two least similar members. Thus, the similarity of two clusters is the similarity of their two most dissimilar members. Complete-link clustering has a similarity function that focuses on global cluster quality. The last similarity function considered is **group-average**. Group average may be considered as a bit of both, single linkage and complete linkage. The criterion for merges is the average similarity between members.

Similarity Measures. As mentioned earlier clustering requires some kind of similarity measure that is computed between objects using the functions described above. Different similarity measures (e.g a good overview is given in [1.13]) and their evaluation [1.2] are available from the statistical natural language processing community . The two most important measures within our work, namely the cosine measure (see Definition 1.3.1) and the kullback leibler divergence (see Definition 1.3.2) are briefly introduced. The cosine measure and the kullback leibler divergence proved to be the most important ones in the area of statistical NLP.

Definition 1.3.1. *The cosine measure or normalized correlation coefficient between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by*

$$cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{x \in X, y \in Y} xy}{\sqrt{\sum_{x \in X} x^2 \sum_{y \in Y} y^2}} \tag{1.1}$$

Using the cosine measure it is computed how well the occurrence of a specific lexical entry correlates in $\boldsymbol{x}$ and $\boldsymbol{y}$ and then divided by the Euclidean length of the two vectors to scale for the magnitude of the individual length of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Though, the following measure is not a metric in the strong sense, it has been quite successfully applied in statistical NLP. The kullback leibler divergence has its roots in information theory and is defined as follows:

Definition 1.3.2. *For two probability mass functions $p(x)$, $q(x)$ their relative entropy is computed by*

$$D(p||q) = \sum_{x \in X} p(x) log \frac{p(x)}{q(x)} \tag{1.2}$$

The kullback leibler divergence is a measure of how different two probability distributions (over the same event space) are. The kullback leibler divergence between p and q is the average number of bits that are wasted by encoding events from a distribution p with a code based on a not-quite-right distribution q. The quantity is always non-negative, and $D(p||q) = 0$ iff $p = q$. An important aspect is that

kullback leibler divergence is not defined for $p(X) > 0$ and $q(x) = 0$. In cases where propability distributions of objects have many zeros, the usage of bottom-up clusering becomes nearly impossible. Thus, for using kullback leibler divergence top-down clustering is the more natural choice.

Example. To explain the similarity measures a small example is given in the following. Imagine a simple concept-concept matrix as given by Table 1.2 consisting of 5 concepts.

Table 1.2. Example Similarity Matrix

ID	HOTEL	ACCOMODATION	ADDRESS	WEEKEND	TENNIS
HOTEL	-	14	7	4	6
ACCOMODATION	14	-	11	2	5
ADDRESS	7	11	-	10	3
WEEKEND	4	2	10	-	5
TENNIS	6	5	3	5	-

Using the cosine measure one may compute the similarity between the concepts HOTEL and ACCOMODATION as follows. The vector of the concept HOTEL is given by $\boldsymbol{x}^{\mathrm{T}} = (0, 14, 7, 4, 6)$, the vector of the concept ACCOMODATION is given by $\boldsymbol{y}^{\mathrm{T}} = (14, 0, 11, 2, 5)$.

$$cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{7 \cdot 11 + 4 \cdot 2 + 6 \cdot 5}{101 \cdot 150} \approx 0.93 \tag{1.3}$$

For computing the Kullback Leibler divergence one has first calculate the probability mass functions for each concept and its corresponding frequencies. The probability mass functions for HOTEL are given as $(0, 0.45, 0.22, 0.13, 0.19)$ the probability mass functions for the concept ACCOMODATION are given as $(0.44, 0, 0.34, 0.06, 0.16)$

Based on these values one can compute the kullback leibler divergence as follows

$$D(\text{HOTEL}||\text{ACCOMODATION}) = 0.22 \cdot \frac{0.22}{0.34} + \ldots + 0.19 \cdot \frac{0.19}{0.16} \approx 0.65 \tag{1.4}$$

We refer the reader to [1.18] where a detailed introduction into further similarity measures between two sets X and Y such as the matching coefficient $X \cap Y$, the dice coefficient $\frac{2|X \cap Y|}{|X|+|Y|}$, the Jaccard or Tanimoto coefficient $\frac{|X \cap Y|}{|X \cup Y|}$ or the overlap coefficient $\frac{|X \cap Y|}{\min(|X|,|Y|)}$ is given.

In the area of information retrieval some work of automatically deriving a hierarchical organization of concepts from a set of documents without use of training data or standard clustering techniques have been presented by Sanderson and Croft [1.22]. They use a subsumption criterion to organize the salient words and phrases

extracted from documents hierarchically. In [1.10] a novel statistical latent class model is used for text mining and interactive information access. In his work the author introduces a Cluster-Abstraction Model (CAM) that is purely data-driven and utilizes context-specific word occurrence statistics. CAM extracts hierarchical relations between groups of documents as well as an abstract organization of keywords.

1.3.2 Statistics-based Refinement of Taxonomic Relations

Statistical data about concepts gathered from text is also used to augment an existing ontology with new concepts. In this case, an automatic classification method is applied to determine a likely place of the new concepts in the taxonomy.

Previous approaches to automatically augmenting an ontology made use of a number of classification techniques which can be summarized according to the following methods: the k nearest neighbor method (kNN), the category-based method and the centroid-based method. They all operate on vector-based semantic representations, which describe the meaning of a word of interest (tar-get word) in terms of counts of its coocurrence with context words, i.e., words appearing within some delineation around the target word. The key differences between the methods stem from different underlying ideas about how a semantic class of words is represented, i.e. how it is derived from the original cooccurrence counts, and, correspondingly, what defines membership in a class. The kNN method is based on the assumption that membership in a class is defined by the new instance's similarity to one or more individual members of the class. Thereby, similarity is defined by a similarity score as, for instance, by the cosine between cooccurrence vectors. To classify a new instance, one determines the set of k training instances that are most similar to the new instance. The new instance is assigned to the class that has the biggest number of its members in the set of nearest neighbors. In addition, the classification decision can be based on the similarity measure between the new instance and its neighbors: each neighbor may vote for its class with a weight proportional to its closeness to the new instance. When the method is applied to augment a thesaurus, a class of training instances is typically taken to be constituted by words belonging to the same synonym set, i.e. lexicalizing the same concept (e.g., [1.8]). A new concept (henceforth - target concept) is assigned in the ontology as a hyponym to that concept (candidate hyperonym) that has the biggest number of its hyponyms among nearest neighbors.

The major disadvantage of the kNN method that is often pointed out is that it involves significant computational expenses to calculate similarity between the new instance and every instance of the training set. A less expensive alternative is the category-based method (e.g., [1.21]). Here the assumption is that membership in a class is defined by the closeness of the new item to a generalized representation of the class. The generalized representation is built by adding up all the vectors constituting a class and normalising the resulting vector to unit length, thus computing a probabilistic vector representing the class. To determine the class of a new word, its unit vector is compared to each vector representing a class. Thus the number of calculations is reduced to the number of classes. Thereby, a class representation may be derived from a set of vectors corresponding to a synonynom set or a set of

vectors corresponding to a synonym set and some or all subordinate synonym sets. In the straightforward case of standard kNN a class is represented by its synonym set. The other extreme is to represent the class in the same manner as is used, e.g., in [1.21] to represent a concept of a thesaurus. Here a class vector would be built from data on all hyponyms of the corresponding concept; new words found similar to this vector would be assigned to that class that lexicalizes the corresponding concept. Another way to prepare a representation of a word class is what may be called the centroid-based approach (e.g., [1.20]). It is almost exactly like the category-based method, the only difference being that the vector representing a class is computed slightly differently. All n vectors corresponding to class members are added up and the resulting vector is divided by n to compute the centroid between the n vectors. As in the case with the category-based approach, one has the same range of choices as to the number of hyponyms of a candidate hyperonym that is used to form a class representation.

1.4 Making use of the structure of the ontology

Existing work in the area of taxonomy learning demonstrated that the symbolic and the statistical approaches have mutually tallying strong points: while the former offers robustness and precision, the latter is characterized by scalability. Accordingly, one can observe a recently increased interest to the hybrid approaches to the task of taxonomy learning (e.g., [1.4]).

In the present paper we will examine possibilities to combine the symbolic approach to represent concepts - the information about the taxonomic organization of an ontology - with the statistical data on them obtained from a corpus in order to increase the accuracy of automatically augmenting the ontology with new concepts.

1.4.1 Tree descending algorithm

One way to factor the taxonomic information into the conceptification decision is to employ the tree-descending" conceptification algorithm, which is a familiar technique in text categorization. The principle behind this approach is that the semantics of every concept in the ontology tree retains some of the semantics of all its hyponyms in such a way that the upper the concept, the more relevant semantic characteristics of its hyponyms it reflects. It is thus feasible to determine the concept of a new word by descending the tree from the root down to a leaf. The semantics of concepts in the ontology tree can be represented by means of one of the three methods to represent a concept described in Section 1.3. At every tree node, the decision which path to follow is made by choosing the child concept that has the biggest distributional similarity to the new word. After the search has reached a leaf, the new word is assigned to that synonym set, which lexicalizes the concept that is most similar to the new word. This manner of search offers two advantages. First, it allows to gradually narrow down the search space and thus save on computational expenses. Second, it ensures that, in a conceptification decision, more

relevant semantic distinctions of potential conceptes are given more preference than less relevant ones.

1.4.2 Tree ascending algorithm

Another way to use information about inter-concept relations contained in an ontology is to base the conceptification decision on the combined measures of distributional similarity and taxonomic similarity (i.e., semantic similarity induced from the hierarchical organization of the ontology) between nearest neighbors. Suppose words in the nearest neighbors set for a given new word, e.g., *trailer*, all belong to different conceptes as in the following conceptification scenario: *box* (similarity score to *trailer*: 0.8), *house* (0.7), *barn* (0.6), *villa* (0.5) (Figure 1.1). In this case, the kNN method will conceptify *trailer* into the concept CONTAINER, since it appears to have biggest similarity to *box*. However, it is obvious that the most likely concept of *trailer* is in a different part of the ontology: in the nearest neighbors set there are three words which, though not belonging to one concept, are semantically close to each other. It would thus be safer to assign the new word to a concept that subsumes one or all of the three semantically similar neighbors. For example, the concepts DWELLING or BUILDING could be feasible candidates in this situation.

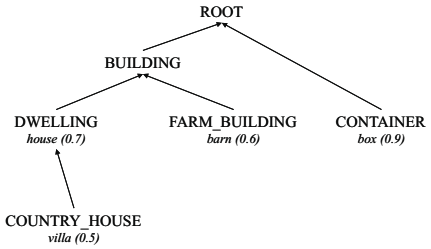

Fig. 1.1. A semantic conceptification scenario

The crucial question here is how to calculate the voting weight for these two concepts to be able to decide which of them to choose or whether to prefer the concept of *box*. Clearly, one cannot sum or average the distributional similarity measures of neighbors below a candidate concept. In the first case the root will always be the best-scoring concept. In the second case the score of the candidate concept will always be smaller than the score of its biggest-scoring hyponym. We propose to estimate the voting weight for such candidate concepts based on taxonomic similarity between relevant nodes. The taxonomic similarity between two concepts is measured according to the procedure elaborated in [1.15]. Assuming that a taxonomy is given as a tree with a set of nodes N, a set of edges $E \subset N \times N$, a unique root ROOT $\in N$, one first determines the least common superconcept of a pair of concepts a, b being compared.

It is defined by

$$\text{lcs}(a,b) := \iota c \in N : \delta(a,c) + \delta(b,c) + \delta(root,c) \text{ is minimal} \tag{1.5}$$

where $\delta(a,b)$ describes the number of edges on the shortest path between a and b. The taxonomic similarity between a and b is then given by

$$\Omega(a,b) := \frac{\delta(\text{ROOT},c)}{\delta(a,c) + \delta(b,c) + \delta(\text{ROOT},c)} \tag{1.6}$$

where $c = lcs(a,b)$. Ω is such that $0 \leq \Omega \leq 1$, with 1 standing for the maximum taxonomic similarity. Ω is directly proportional to the number of edges from the least common superconcept to the root, which agrees with the intuition that a given number of edges between two concrete concepts signifies greater similarity than the same number of edges between two abstract concepts.

The first method to calculate the voting weight for a candidate concept is to sum the distributional similarity measures of its hyponyms to the target word t each weighted by the taxonomic similarity measure between the hyponym and the candidate node:

$$W(n) := \Sigma_{h \in I_n} \text{sim}(t,h) \cdot \Omega(n,h) \tag{1.7}$$

where I_n is the set of hyponyms below the candidate concept n, $\text{sim}(t,h)$ is the distributional similarity between a hyponym h and the word to be conceptified t, and $\Omega(n,h)$ is the taxonomic similarity between the candidate concept and the hyponym h.

1.5 Data and Settings of the Experiments

The machine-readable ontology we used in this study was derived from GETESS [1.23], an ontology for the tourism domain. Each concept in the ontology is associated with one lexical item, which expresses this concept. From this ontology, word classes were derived in the following manner. A class was formed by words lexicalizing all child concepts of a given concept. For example, the concept CULTURAL_EVENT in the ontology has successor concepts PERFORMANCE, OPERA, FESTIVAL, associated with words *performance*, *opera*, *festival* correspondingly. Though these words are not synonyms in the traditional sense, they are taken to constitute one semantic class, since out of all words of the ontology's lexicon their meanings are closest. The ontology thus derived contained 1052 words and phrases (out of these, 756 cropped up in the corpus at least once) grouped into 182 classes. The corpus from which distributional data about the words were obtained was extracted from a web site advertising hotels around the world. It contained about 6 megabytes of text (988 000 words).

Collection of distributional data was carried out in the following settings. The preprocessing of corpus included a very simple stemming (most common inflections were chopped off; irregular forms of verbs, adjectives and nouns were changed to

their first forms). The context of usage was delineated by a window of 3 words on either side of the target word, without transgressing sentence boundaries. In case a stop word other than a proper noun appeared inside the window, the window was accordingly expanded. The stoplist included 50 most frequent words of the British National Corpus, words listed as function words in the BNC, and proper nouns not appearing in the sentence-initial position. The obtained frequencies of cooccurrence were weighted by the 1+log weight function. The distributional similarity was measured by means of three different similarity metrics: the Jaccard's coefficient, L1 distance, and the skew divergence, a weighted version of the Kullback-Leibler divergence (cf., [1.14]).

1.6 Evaluation method

The performance of the algorithms was assessed in the following manner. For each algorithm, we held out a single word of the ontology as the test case, and trained the system on the remaining 755 words. We then tested the algorithm on the held-out vector, observing if the assigned concept for that word coincided with its original concept in the ontology, and counting the number of correct conceptifications ("direct hits"). This was repeated for each of the words of the ontology.

However, given the intuition that a semantic conceptification may not be simply either right or wrong, but rather of varying degrees of appropriateness, we believe that a clearer idea about the quality of the conceptifiers would be given by an evaluation method that takes into account "near misses" as well. We therefore evaluated the performance of the algorithms also in terms of how close the proposed concept for a test word was to the correct concept. For this purpose we measured the taxonomic similarity between the assigned and the correct conceptes of words so that the appropriateness of a particular conceptification was estimated on a scale between 0 and 1, with 1 signifying assignment to the correct concept. Thus this measure of accuracy of conceptifications was compatible with the counting of direct hits, which, as will be shown later, may be useful for evaluating the methods. In the following, the evaluation of the conceptification algorithms is reported both in terms of the average of direct hits and the average of the taxonomy similarity between the assigned and the correct conceptes ("direct+near hits") over all words in the ontology.

To have a benchmark for evaluation of the algorithms, a baseline was calculated, which was the average hit value a given word gets, when its concept label is chosen at random. The baseline for direct hits was estimated at 0.012; for direct+near hits, it was 0.15741.

1.7 Results

We first conducted experiments evaluating performance of the three standard classifiers. To determine the best version for each particular classifier, only those parameters were varied that, as described above, we deemed to be critical for a specific

algorithm in the setting of ontology augmentation. Other parameters can have a serious impact on the absolute quality of their performance, but we, however, were interested in comparing the classifiers relative to each other.

In order to get a view on how the accuracy of the algorithms was related to the amount of available distributional data on the target word, all words of the ontology were divided into three groups depending on the amount corpus data available on them (see Table 1.3). The amount of distributional data for a word (the frequency in the left column) is the total of frequencies of its context words.

Table 1.3. Distribution of words of the ontology into frequency ranges

Frequency range	# words in the range
0-40	274
40-500	190
>500	292

The results of the evaluation of the methods are summarized in the tables below. Rows specify the metrics used to measure distributional similarity and columns specify frequency ranges. Each cell describes the average of direct+near hits / the average of direct hits over words of a particular frequency range and over all words of the ontology. The statistical significance of the results was measured in terms of the one-tailed chi-square test.

kNN. The evaluation of the method was conducted with $k = 1, 3, 5, 7, 10, 15, 20, 25,$ and 30. The accuracy of conceptifications increased with the increase of k. However, starting with $k = 20$ the increase of k yielded only insignificant improvement. Table 1.4 describes results of evaluation of kNN using 30 nearest neighbors, which was found to be the best version of kNN.

Table 1.4. kNN, k=30[4]

	0-40	40-500	>500	Overall
JC	.33773 /.17142	.33924 /.15384	.40181 /.12457	.37044 /.15211
L1	.33503 /.16428	.38424 /.21025	.38987 /.14471	.37636 /.17195
SD	.31505 /.14285	.36316 /.18461	.45234 /.17845	.38806 /.17063

Category-based method. To determine the best version of this method, we experimented with the number of levels of hyponyms below a concept that were used to

[3]

Acronym	Full name
JC	Jacquard Coefficient
L1	L1 Metric
SD	Skew Divergence

build a concept vector). The best results were achieved when a concept was represented by data from its hyponyms at most three levels below it (Table 1.5).

Table 1.5. Category-based method, 3 levels

	0-40	40-500	>500	Overall
JC	.26918 /.12142	.34743 /.17948	.47404 /.28282	.37554 /.2023
L1	.27533 /.125	.41736 /.25128	.56711 /.38383	.43242 /.26190
SD	.28589 /.12857	.34932 /.18461	.51306 /.31649	.39755 /.21957

Centroid-based method. As in the case with the category-based method, we varied the number of levels of hyponyms below the candidate concept. Table 1.6 details results of evaluation of the best version of this method (a concept is represented by 3 levels of its hyponyms).

Table 1.6. Centroid-based method, 3 levels

	0-40	40-500	>500	Overall
JC	.17362 /.07831	.18063 /.08119	.30246 /.14434	.22973 /.10714
L1	.21711 /.09793	.30955 /.13938	.37411 /.1687	.30723 /.12698
SD	.22108 /.09972	.23814 /.11374	.36486 /.16147	.28665 /.10714

Comparing the three algorithms we see that overall, kNN and the category-based method exhibit comparable performance (with the exception of measuring similarity by L1 distance, when the category-based method outperforms kNN by a margin of about 5 points; statistical significance $p < 0.001$). However, their performance is different in different frequency ranges: for lower frequencies kNN is more accurate (e.g., for L1 distance, $p < 0.001$). For higher frequencies, the category-based method improves on kNN (L1, $p < 0.001$). The centroid-based method exhibited performance, inferior to both those of kNN and the category-based method.

Tree descending algorithm. In experiments with the algorithm, candidate concepts were represented in terms of the category-based method, 3 levels of hyponyms, which proved to be the best generalized representation of a concept in previous experiments. Table 1.7 specifies the results of its evaluation.

Table 1.7. Tree descending algorithm

	0-40	40-500	>500	Overall
JC	.00726 /0	.01213 /.00512	.02312 /.0101	.014904 /.005291
L1	.08221 /.03214	.05697 /.02051	.21305 /.11111	.128844 /.060846
SD	.08712 /.03214	.07739 /.03589	.16731 /.06734	.011796 /.047619

Its performance turns out to be much worse than that of the standard methods. Both direct+near and direct hits scores are surprisingly low, for 0-40 and 40-500 much lower than chance. This can be explained by the fact that some of top concepts in the tree are represented by much less distributional data than other ones. For example, there are less than 10 words that lexicalize the top concepts MASS_CONCEPT and MATHEMATICAL_CONCEPT and all of their hyponyms (compare to more than 150 words lexicalizing THING and its hyponyms up to 3 levels below it). As a result, at the very beginning of the search down the tree, a very large portion of test words was found to be similar to such concepts.

Tree ascending algorithm. The experiments were conducted with the same number of nearest neighbors as with kNN. Table 1.8 describes the results of evaluation of the best version (equation 1.7, $k = 15$).

Table 1.8. Tree ascending algorithm, voting weight according to equation 1.7, k=15.

	0-40	40-500	>500	Overall
JC	.32112 /.075	.33553 /.0923	.40968 /.08754	.36643 /.08597
L1	.33369 /.07142	.34504 /.0923	.42627 /.09764	.38005 /.08862
SD	.31809 /.06785	.32489 /.05128	.45529 /.11111	.38048 /.08201

There is no statistically significant improvement on kNN overall, or in any of the frequency ranges. The algorithm favored more upper concepts and thus produced about twice as few direct hits than kNN. At the same time, its direct+near hits score was on par with that of kNN! This algorithm thus produced much more near hits than kNN, what can be interpreted as its better ability to choose a superconcept of the correct concept. Based on this observation, we combined the best version of the tree ascending algorithm with kNN in one algorithm in the following manner. First the former was used to determine a superconcept of the concept for the new word and thus to narrow down the search space. Then the kNN method was applied to pick a likely concept from the hyponyms of the concept determined by the tree ascending method. Table 1.9 specifies the results of evaluation of the proposed algorithm.

Table 1.9. Tree ascending algorithm combined with kNN, k=30

	0-40	40-500	>500	Overall
JC	.34444 /.16428	.35858 /.14358	.41260 /.10774	.38215 /.14021
L1	.35147 /.16428	.36545 /.15384	.41086 /.11784	.38584 /.14682
SD	.32613 /.13571	.36485 /.1641	.45732 /.16498	.39456 /.1574

The combined algorithm demonstrated improvement both on kNN and the tree ascending method of 1 to 3 points in every frequency range and overall for direct+near hits (except for the 40-500 range, L1). The improvement was statistically significant only for L1, > 500 ($p = 0.05$) and for L1, overall ($p = 0.011$). For other

similarity measures and frequency ranges it was insignificant (e.g., for JC, overall, $p = 0.374$; for SD, overall, $p = 0.441$). The algorithm did not improve on kNN in terms of direct hits. The hits scores set in bold in Table 1.9 are those which are higher than those for kNN in corresponding frequency ranges and similarity measures.

1.8 Conclusion

In this section we have surveyed symbolic and statistical means for extracting taxonomic relations from text. We have proposed new algorithms that combine the advantages of scalable statistical approaches with symbolic approaches that consider background knowledge.

For this proposal we have shown how to evaluate learning approaches in a typical web setting. Though these first steps have not yet resulted in a significant improvement we may conjecture that with the further study of improvement of the combination of of combined symbolic and statistical approaches we may arrive at a good quality conceptification of unknown words. In particular, the aggregation of such multiple approaches will yield a basis for multi-strategy learning as well as for estimating the reliability of learned taxonomies — which is necessary in order to embed these algorithms into actual ontology development environments like [1.25].

In particular, our study demonstrated that taxonomic similarity between nearest neighbors, in addition to their distributional similarity to the new word, may be a useful evidence on which conceptification decision can be based. We have proposed a "tree ascending" conceptification algorithm which extends the kNN method by making use of the taxonomic similarity between nearest neighbors. This algorithm was found to have a very good ability to choose a superconcept of the correct concept for a new word. On the basis of this finding, another algorithm was developed that combines the tree ascending algorithm and kNN in order to optimize the search for the correct concept. Although only limited statistical significance of its improvement on kNN was found, the results of the study indicate that this algorithm is a promising possibility to incorporate the structure of a ontology into the decision as to the concept of the new word. In particular, we conjecture that the tree ascending algorithm leaves a lot of room for improvements and combinations with other algorithms like kNN. The tree descending algorithm, a technique widely used for text categorization, proved to be much less efficient than standard conceptifiers when applied to the task of augmenting a domain-specific ontology. Its poor performance is due to the fact that in such a ontology there are great differences between top concepts in the amount of distributional data used to represent them.

In order to have a better understanding of the role of different parameters on the performance of these conceptifiers, they can be further studied on the material of a large-scale ontology, where richer information about its structural organization is available. Also study on further domains like organizational memories or genomics let us expect fruitful results with regard to methods and applications. We anticipate that with further evaluation of combined symbolic/statistic approaches we may

arrive at a good quality conceptification of unknown concepts. In particular, the aggregation of several such approaches will yield a basis for multi-strategy learning as well as for estimating the reliability of learned taxonomies - which is necessary for speedy and efficient maintaining ontologies for the Web.

References

1.1 H Assadi. Construction of a regional ontology from text and ist use within a documentary system. In *N. Guarino (ed.), Formal Ontology in Information Systems, Proceedings of FOIS-98, Trento, Italy, 1999*, pages 236–249, 1999.

1.2 I. Dagan, L. Lee, and F. Pereira. Similarity-based models of word coocurrence probabilities. *Machine Learning*, 34(1):43–69, 1999.

1.3 D. Faure and C. Nedellec. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In *In LREC workshop on Adapting lexical and corpus resources to sublanguages and applications, Granada, Spain, Mai 1998.*, 1998.

1.4 G. Greffenstette and M. A. Hearst. A method for refining automatically-discovered lexical relations: Combining weak techniques for stronger results. In *Proceedings of the Workshop on Statistically-Based Natural Language Programming Techniques, AAAI Press, Menlo Park, CA, 1992*, 1992.

1.5 U. Hahn and Schnattinger. Ontology engineering via text understanding. In *IFIP'98 — Proceedings of the 15th World Computer Congress*, Vienna and Budapest, 1998.

1.6 Peter M. Hastings. *Automatic Acquisition of Word Meaning from Context*. PhD thesis, University of Michigan, 1994.

1.7 M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics. Nantes, France*, 1992.

1.8 Marti A. Hearst and Hinrich Schütze. Customizing a lexicon to better suit a computational task. In *Proc. of the ACL-SIGLEX Workshop on Acquisition of Lexical Knowledge from Text, Columbus, Ohio, USA*, 1993.

1.9 J. Hobbs. The generic information extraction system. In *Proceedings of the Fifth Message Understanding Conference (MUC-5), Morgan Kaufmann, 1993.*, 1993.

1.10 T. Hofmann. The Cluster-Abstraction Model: Unsupervised Learning of Topic Hierarchies from Text Data. In *Proceedings of 16th International Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden, 1999*, pages 682–587, 1999.

1.11 L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

1.12 J.-U. Kietz, R. Volz, and A. Maedche. Semi-automatic ontology acquisition from a corporate intranet. In *International Conference on Grammar Inference (ICGI-2000)*, to appear: Lecture Notes in Artificial Intelligence, LNAI, 2000.

1.13 L. Lee. Measures of distributional similarity. In *Proceedings of the ACL'99*, pages 25–32, 1999.

1.14 Lilian Lee. Measures of distributional similarity. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, 1999.

1.15 A. Maedche and S. Staab. Discovering conceptual relations from text. In *ECAI-2000 - European Conference on Artificial Intelligence. Proceedings of the 13th European Conference on Artificial Intelligence*. IOS Press, Amsterdam, 2000.

1.16 A. Maedche and S. Staab. Mining ontologies from text. In *Proceedings of EKAW-2000, Springer Lecture Notes in Artificial Intelligence (LNAI-1937), Juan-Les-Pins, France, 2000*. Springer, 2000.

1.17 A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2), 2001.

1.18 C.D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.
1.19 E. Morin. Automatic acquisition of semantic relations between terms from technical corpora. In *Proc. of the Fifth International Congress on Terminology and Knowledge Engineering - TKE'99*, 1999.
1.20 F. Pereira, N. Tishby, and L. Lee. Distributation Clustering of English Words. In *Proceedings of the ACL-93, 1993*, pages 183–199, 1993.
1.21 Philip S. Resnik. *Selection and Information: A Class-based Approach to Lexical Relationships*. PhD thesis, University of Pennsylania, 1993.
1.22 M. Sanderson and B. Croft. Deriving Concept Hierarchies from Text. In *Proceedings of the International Conference on Information Retrieval — SIGIR'99, August 1999, Berkley CA, USA*, 1999.
1.23 S. Staab, C. Braun, A. Düsterhöft, A. Heuer, M. Klettke, S. Melzig, G. Neumann, B. Prager, J. Pretzel, H.-P. Schnurr, R. Studer, H. Uszkoreit, and B. Wrenger. GETESS — searching the web exploiting german texts. In *Proceedings of the 3rd Workshop on Cooperative Information Agents*, LNCS-1652, Berlin, 1999. Springer.
1.24 S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1), 2001.
1.25 York Sure, Michael Erdmann, Juergen Angele, Steffen Staab, Rudi Studer, and Dirk Wenke. Ontoedit: Collaborative ontology development for the semantic web. In *Proceedings of the 1st International Semantic Web Conference (ISWC2002), June 9-12th, 2002, Sardinia, Italia*, LNCS. Springer, 2002.