Learning Concept Hierarchies from Text with a Guided Hierarchical Clustering Algorithm

Philipp Cimiano CIMIANO@AIFB.UNI-KARLSRUHE.DE
Institute for Applied Computer Science and Formal Methods, University of Karlsruhe

Steffen Staab STAAB@UNI-KOBLENZ.DE
Institute for Computer Science, University of Koblenz-Landau

Abstract

We present an approach for the automatic induction of concept hierarchies from text collections. We propose a novel guided agglomerative hierarchical clustering algorithm exploiting a hypernym oracle to drive the clustering process. By inherently integrating the hypernym oracle into the clustering algorithm, we overcome two main problems of unsupervised clustering approaches relying on the distributional similarity of terms to induce concept hierarchies. First, by only clustering two terms if they have a hypernym in common we make sure that the cluster produced in this way is actually reasonable. Second, by labeling the clusters with the corresponding hypernym we overcome the labeling problem shared by all unsupervised approaches. We present results of a comparison of our approach with Caraballo's method, assessing the quality of the automatically learned ontologies by comparing them to a handcrafted taxonomy for the tourism domain using the similarity measures of Maedche et al. Further, we also present a human evaluation of the concept hierarchy produced by our guided algorithm.

1. Introduction

Most approaches aiming at learning concept hierarchies are based on unsupervised learning paradigms. These approaches rely on the possibility of assessing the semantic similarity between words on the basis of the amount of linguistic context they share in a given corpus (compare [9]). In order to induce a hierarchy between concepts, many approaches exploit clustering algorithms such as the approach in [13] using a soft clustering method relying on deterministic annealing to find lowest distortion sets of clusters. Others use agglomerative clustering [1, 2, 7] as well as divisive algorithms such as Bi-Section-KMeans [3] or conceptual clustering algorithms such as Formal Concept Analysis [3]. However, there are two major problems shared by these approaches. On the one hand, there is the problem of data sparseness leading to the fact that certain syntactic similarities with respect to the corpus are accidental and due to missing data (cf. [18]), thus not corresponding to real-world or semantic similarities. On the other hand all the approaches share the problem of not being able to appropriately label the produced clusters. In this paper we present a new algorithm addressing both issues. The algorithm is a novel guided hierarchical agglomerative clustering algorithm exploiting a hypernym oracle automatically extracted from different resources in a first step. Though our approach also makes use of hypernyms extracted by other means for labeling the concepts as in [2], the principle difference is that instead of merely postprocessing the hierarchy, in our approach the hypernyms are directly used to guide the clustering algorithm. In fact, in our guided algorithm two terms are only clustered if there is a corresponding common hypernym according to the oracle, thus making the clustering less error-prone. We demonstrate this claim by presenting results comparing our approach with Caraballo's algorithm on a tourism-related dataset. Further, we also present a human evaluation of the concept hierarchy produced by our guided algorithm. The paper is structured as follows: the following Section 2 describes the guided agglomerative clustering algorithm. Section 3 presents the evaluation of the approach and Section 4 discusses some related work. Section 5 concludes.

2. Oracle-Guided Agglomerative Clustering

In this section we present the guided agglomerative clustering approach for learning concept hierarchies. The approach relies on the distributional similarity of terms with respect to an underlying corpus. Furthermore, it is guided in the sense that it exploits hypernyms acquired by other means to drive the clustering process. In particular, the

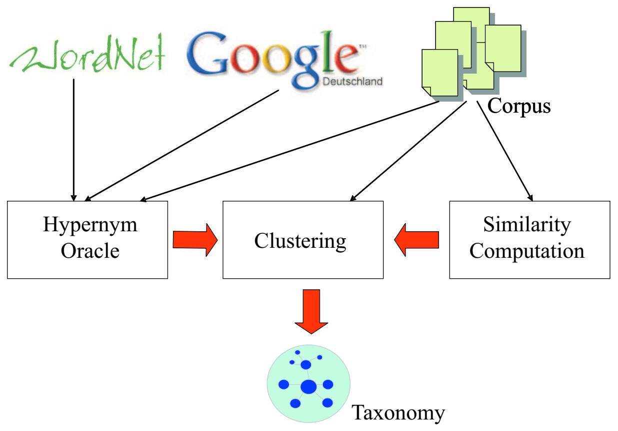

Figure 1. System Overview

approach exploits hypernyms extracted from WordNet as well as an approach matching lexico-syntactic patterns indicating a hypernym-relationship as suggested in [10]. The clustering algorithm is then driven by these extracted hypernyms in the sense that given two terms which are similar according to their corpus behavior, it will either order them as subconcepts, in case one is a hypernym of the other, or – in case they have a common hypernym – add them as sisters under a concept labeled with that hypernym. Figure 1 gives an overview of the system. In what follows we first describe how the similarity between terms is calculated in Section 2.1. Then we describe our method for extracting hypernyms from different resources in Section 2.2. After presenting the actual algorithm in Section 2.3 we discuss an example for illustration purposes in Section 2.4.

2.1. Calculating Term Similarities

In order to calculate the similarity between terms, we rely on Harris' distributional hypothesis [9] claiming that terms are semantically similar to the extent to which they share similar syntactic contexts. For this purpose, for each term in question we extract syntactic surface dependencies from the corpus. These surface dependencies are extracted by matching regular expressions over part-of-speech tags. In what follows we list the syntactic expressions we use and give a brief example of how the features, represented as predicates, are extracted from these expressions:

- adjective modifiers, i.e. *a nice city* → nice(city)
- prepositional phrase modifiers, i.e. *a city near the river* → near_river(city) and city_near(river), respectively
- possessive modifiers, i.e. *the city's center* → has_center(city)
- noun phrases in subject or object position. i.e. *the city offers an exciting nightlife* → offer_subj (city) and offer_obj(nightlife)
- prepositional phrases following a verb, i.e. *the river flows through the city* → flows_through(city)
- copula constructs i.e. *a flamingo is a bird* → is_bird(flamingo)
- verb phrases with the verb *to have*, i.e. *every country has a capital* → has_capital(country)

Consider for example the following discourse:

Mopti is the biggest city along the Niger with one of the most vibrant ports and a large bustling market. Mopti has a traditional ambience that other towns seem to have lost. It is also the center of the local tourist industry and suffers from hard-sell overload. The nearby junction towns of Gao and San offer nice views over the Niger's delta.

Here we would extract the following concept vectors for each object, where the number in paranthesis gives the absolute frequency for each feature:

city:	biggest(1)
ambience:	traditional(1)
center:	of_tourist_industry(1)
junction towns:	nearby(1)
market:	bustling(1)
port:	vibrant(1)
overload:	suffer_from(1)
tourist industry:	center_of(1), local(1)
town:	seem_subj(1)
view:	nice(1), offer_obj(1)

On the basis of these vectors we calculate the similarity between two terms t_1 and t_2 as the cosine between their corresponding vectors:

$$\cos(\sphericalangle(\vec{t}_1, \vec{t}_2)) = \frac{\vec{t}_1 \cdot \vec{t}_2}{\parallel \vec{t}_1 \parallel \cdot \parallel \vec{t}_2 \parallel}$$

According the the above cosine measure, the following ten pairs of terms are the ten most similar terms of the reference taxonomy with respect to our corpus (compare the dataset description in Section 3):

(t_1,t_2)	Sim
(autumn,summer)	0.93
(autumn,night)	0.83
(summer,spring)	0.72
(person,living_thing)	0.69
(trip,visit)	0.68
(winter,summer)	0.66
(badminton,tennis)	0.65
(day,morning)	0.64
(tennis,golf)	0.64
(farm,town)	0.62

2.2. The Hypernym Oracle

The guided agglomerative clustering algorithm relies on an oracle returning possible hypernyms for a given term. Thus, before applying the actual algorithm, the oracle needs to be constructed. In this section we describe an automatic approach to construct such an oracle which in essence is a function

$$H : String \rightarrow 2^{String \times \mathbb{N}}$$

which for a term t returns a set of tuples (h, f), where h is a hypernym and f is the number of times the algorithm has found evidence for it. We also define the first projection $H_1(t)$ returning the set of hypernyms of t:

$$H_1(t) := \{h \mid \exists n(h, n) \in H(t)\}$$

In order to find these hypernyms we make use of three sources: (i) WordNet, (ii) Hearst patterns matched in a corpus and (iii) Hearst patterns matched in the World Wide Web (compare [4]). WordNet [8] is a lexical database in which terms are organized in so-called synsets consisting of synonyms and thus representing a specific meaning of a given term. For each term t we thus collect all the hypernyms in the synsets which dominate any synset in which t appears. We add these hypernyms to $H(t)$ together with the number of times that the corresponding hypernym appears in a dominating synset.
Furthermore, we also apply the lexico-syntactic patterns described in [10] to find hypernyms in the underlying corpus. The following patterns we use are taken from [10][1]:

(1) NP_0 such as NP_1, NP_2, ..., NP_{n-1} (and|or) NP_n
(2) such NP_0 as NP_1, NP_2, ... NP_{n-1} (and|or) NP_n
(3) NP_1, NP_2, ..., NP_n (and|or) other NP_0
(4) NP_0, (including|especially) NP_1, NP_2, ..., NP_{n-1} (and|or) NP_n

According to Hearst, from the above patterns we can derive that for all NP_i, $1 \leq i \leq n$, $is\text{-}a(head(NP_i), head(NP_0))$. In addition, we also use the following patterns:

(5) NP_1 is NP_0
(6) NP_1, another NP_0
(7) NP_0 like NP_1

Now given two terms t_1 and t_2 we record how many times a Hearst-pattern indicating an *isa*-relation between t_1 and t_2 is matched in the corpus. In order to match the above patterns we create regular expressions over part-of-speech tags to match NP's. In particular, we use the tagger described in [16] and match non-recursive NP's consisting of a determiner, an optional sequence of modifying adjectives and a sequence of common nouns constituting the head of the NP. Additionally, we also follow an approach in which web pages are actually downloaded and Hearst patterns are matched offline. For this purpose, we assign one or more functions $f_i : string \rightarrow string$ – which we will refer to as *clues* – to each of the Hearst patterns i to be matched. Given a concept of interest c, we instantiate each of the clues and download a number of pages matching the query $f_i(c)$ using the Google API. For example, given the clue $f(x) = "such\ as" \oplus \pi(x)$ and the concept *conference* we would download the first 100 Google abstracts matching the query f(conference), i.e. "such as conferences".[2] For each concept of interest and for each of the correspondingly instantiated clues, we then process the downloaded documents by matching the corresponding pattern, thus yielding its potential superconcepts. The following table gives the clues used as well as the corresponding Hearst patterns:

Clue	Hearst pattern
$f(x) = "such\ as" \oplus \pi(x)$	(1)
$f(x) = \pi(x) \oplus "and\ other"$	(3)
$f(x) = \pi(x) \oplus "or\ other"$	(3)
$f(x) = "including" \oplus \pi(x)$	(4)
$f(x) = "especially" \oplus \pi(x)$	(4)
$f(x) = x \oplus "is"$	(5)

The following table for example shows the results of the above described hypernym extraction process for the term *summer*. In particular, for each resource it gives the hypernyms found as well as the number of times such an evidence was found in the corresponding resource:

[1] NP_i stands for a noun phrase.

[2] Here, $\oplus$ denotes the concatenation operator defined on two strings and $\pi(t)$ is a function returning the correct plural form of t.

Hearst Corpus	
is-a(summer,heat)	1
is-a(summer,performer)	1
is-a(summer,time)	1
is-a(summer,mind)	1
is-a(summer,tubing)	1
Hearst WWW	
is-a(summer,time)	3
is-a(summer,vacation)	2
is-a(summer,period)	1
is-a(summer,season)	1
is-a(summer,skill)	1
WordNet	
is-a(summer,period)	1

In this example, the results of the different resources would add up to 4 for the hypernym *time*, 2 for the hypernym *vacation* and 2 for the hypernym *period* as well as 1 for the rest of the candidate hypernyms in the table.

2.3. Algorithm

In this section we describe the guided agglomerative clustering algorithm for inducing concept hierarchies. The algorithm is given by the following pseudocode:

```
1. Input: a list T of $n$ terms to be ordered hierarchically

2. calculate the similarity between each pair of terms ($O(n^2)$) and sort them from highest to lowest ($O(n\ log\, n)$)
   initialize the set of clustered terms C, i.e. C:={}

3. FOREACH pair $(t_1, t_2)$ in the ordered list representing a potential pair to be clustered, if either $t_1$ or $t_2$ has not been NOT classified as subconcept of some other concept:
   (a) IF $(t_1, m) \in H(t_2)$
       i. IF $(t_2, n) \in H(t_1)$ and $n > m$, then isa($t_1$,$t_2$)
       ii. ELSE isa($t_2$,$t_1$)
   (b) ELSE IF $(t_2, m) \in H(t_1)$
       i. isa($t_2$,$t_1$)
   (c) ELSE IF $(h, n) \in H(t_1)$ and $(h, m) \in H(t_2)$ and there is no $h'$ such that $(h', p) \in H(t_1)$ and $(h', q) \in H(t_2)$ and $p + q > m + n$
       i. IF isa($t_1$,$t'$), i.e $t_1$ is already classified as $t'$
          A. IF t' == h, then isa($t_2$,$t'$)
          B. ELSE IF $(h, n) \in H(t')$ and $((t', m) \in H(h) \rightarrow m < n)$
             IF $t_2$ has not yet been classified, then isa($t_2$,$t'$)
             IF $t'$ has not yet been classified, then isa($t', h$)
          C. ELSE
             IF $t_2$ has not yet been classified then isa($t_2$,$h$)
             IF $h$ has not yet been classified, then isa($h, t'$)
       ii. ELSE IF isa($t_2$,$t'$), i.e. $t_2$ is already classified as $t'$ (analogous case to 3c i)
          A. IF t' == h, then isa($t_1$,$t'$)
          B. ELSE IF $(h, n) \in H(t')$ and $((t', m) \in H(h) \rightarrow m < n)$
             as $t_1$ has not yet been classified, then isa($t_1$,$t'$)
             IF $t'$ has not yet been classified, then isa($t', h$)
          C. ELSE
             as $t_1$ has not yet been classified, then isa($t_1$,$h$)
             IF $h$ has not yet been classified, then isa($h, t'$)
       iii. ELSE, as neither $t_1$ nor $t_2$ have been classified, isa($t_1$,$h$), isa($t_2$,$h$) .
   (d) ELSE, as there are no common hypernyms, mark $t_1$ and $t_2$ as clustered, i.e. C: = C $\cup$ ($t_1$,$t_2$)

4. FOREACH term $t \in T$ which has not been processed (because no similar terms were found in the corpus), if there is some other term $t'$ in C such that substringOf(t',t), then isa($t$,$t'$)

5. FOREACH ($t_1$,$t_2$) $\in C$
   (a) IF there is a $t'$ such that isa($t_1$,$t'$) THEN isa($t_2$,$t'$)
   (b) ELSE IF there is a $t'$ such that isa($t_2$,$t'$) then isa($t_1$,$t'$)
   (c) ELSE select the pair $(t', m) \in H(t_1) \cup H(t_2)$ for which there is no $(t'', n) \in H(t_1) \cup H(t_2)$ such that $n > m$ and create the following structures: isa($t_1$,$t'$) and isa($t_2$,$t'$)

6. FOREACH term $t \in T$ which has not been classified, put it directly under the top concept, i.e. isa($t$,top)

7. Output: a labeled concept hierarchy for the words in T
```

For each pair (t_1, t_2) the algorithm thus first consults the hypernym oracle to find out if t_1 is a hypernym of t_2 or the other way round, creating the appropriate subconcept relation (3a and 3b). If this is not the case (3c), it consults the oracle for common hypernyms of both terms, selecting the most frequent hypernym h and distinguishes three cases. In the case none of the terms has been already classified (3c iii), it creates a new concept labeled with h together with two subconcepts labeled as t_1 and t_2. In case one of the two terms, say t_1, has been already classified as $isa(t_1, t')$, there are three more cases to distinguish. In the first case (3c i.A), if h and t' are identical, the algorithm simply puts a concept t_2 under t' (compare Figure 2 (left)). In the second case (3c i.B), if according to the oracle h is a hypernym of t', it creates the structure in Figure 2 (middle). In case it is not an hypernym (3c i.C), it creates the structure in Figure 2 (right). The algorithm proceeds analogously in case t_2 has been already classified. In case there are no common hypernyms t_1 and t_2 are simply marked as clustered for further processing (3d). This is done for all the similarity pairs provided that one of the two terms has not been classified yet.

After this process, the algorithm exploits the vertical-relations heuristic in [17] adding t_1 as subconcept of t_2 if t_2 is a substring t_1 in the way *credit card* is a substring of *international credit card* (4). Then, all the pairs (t_1,t_2) which have been clustered and kept for later processing are considered (5), and if either t_1 or t_2 has been already classified (5a and 5b), the other term is added under the corresponding superconcept. If this is not the case, both terms are added as subconcepts of the most frequent hypernym in $H(t_1) \cup H(t_2)$ according to the oracle (5c). At the end, every unclassified term is added directly under the top concept (6).

In general, each time an *isa*-relation is added, the algorithm has to check that no cycles are created by introducing the relation in question. The overall time complexity of the algorithm is thus $O(n^2)$ as steps 2, 3, 4 and 5 have complexity $O(n^2)$ and step 6 is even linear in the number of terms $n = |T|$. The algorithm is thus as efficient as agglomerative clustering with single-linkage and more efficient than agglomerative clustering with complete and average linkage (compare [5])[3].

As already mentioned in the introduction this algorithm can be considered as guided as it depends on an external hypernym oracle. The obvious benefit is that by only clustering terms in case they have a common hypernym according to the oracle, the clustering process is more controlled and less error-prone. This claim is demonstrated experimentally in Section 3. Furthermore, the approach also allows to label abstract concepts in an appropriate way.

It is important to emphasize that the outcome of the algorithm does not simply mirror the WordNet hierarchy, but is in fact implicitly performing sense disambiguation. In fact, due to the fact that we look up the common hypernym of two terms which are similar with respect to the underlying corpus, we are more likely to find a hypernym (of the many contained in WordNet for both terms separately) which corresponds to the common sense of both terms in the domain in question thus finding more appropriate labels than when processing each term separately.

[3]See also http://www-csli.stanford.edu/∼schuetze/ completelink.html on this issue

(t_1,t_2)	Sim	Hypernym	Count
(autumn,summer)	0.93	period	3
(autumn,night)	0.83	period	5
(summer,spring)	0.72	period	3
(person,living_thing)	0.69		
(trip,visit)	0.68	activity	23
		event	10
		travel	3
		outing	2
(winter,summer)	0.66	season	3
(badminton,tennis)	0.65	human_activity	2
		sport	2
(day,morning)	0.64	time	10
		period	9
		day	4
		work	4
		others	2
(tennis,golf)	0.64	sport	2
(farm,town)	0.62	area	15
		place	9
		entity	6
		landscape	6
		unit	5
		country	2
		structure	2

Table 1. Common Hypernyms with occurrences for the top ten most similar pairs of terms

2.4. An Example

In order to illustrate the above algorithm, consider again the top ten most similar pairs according to a collection of tourism-related texts (see section 3 for details about the dataset) together with their common hypernyms as well as the corresponding occurrences in Table 1.

After the first three steps of the algorithm, *autumn*, *summer*, *night* and *spring* will have been added as subconcepts of a concept labeled with *period* according to steps 3c iii, 3c i.A and 3c i.A, respectively. In the fourth step, as *living_thing* is a hypernym of *person* according to our hypernym oracle, *person* is added as a subconcept of *living_thing* according to case 3a of our algorithm. In the 5th step, *trip* and *visit* are added as subconcepts of a concept labeled with *activity* according to step 3c iii. Interesting is the 6th step, in which, as *season* is a hypernym of *period* following the oracle, according to case 3c ii.B a new concept labeled with *season* is created with *period* and *winter* as subconcepts. Then, *badminton* and *tennis* are added as subconcepts of *human_activity* according to case 3c iii. In the 8th step, according again to case 3c iii., a new concept *time* is created

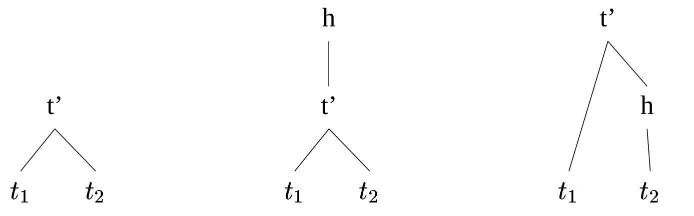

Figure 2. Structures constructed in algorithm steps 3c i.A and 3c i.B and 3c i.C, respectively

with *day* and *morning* as subconcepts. Finally, as *sport* is a *human_activity*, *golf* is added as a subconcept of *sport* according to step 3c i.B; *farm* and *town* are added as subconcepts of a new concept *area* according to 3c iii.
As all pairs have been processed, then as *activity* is a substring *human_activity*, the latter, following step 4, is added as a subconcept of the former thus yielding at the end the concept hierarchy depicted in Figure 2.4. This hierarchy is certainly far from perfect but also shows that the results of our algorithm are quite reasonable.

3. Evaluation

In order to evaluate the automatically produced concept hierarchies, we compare them to a handcrafted reference concept hierarchy but also present the hierarchy to a human subject in order to assess its quality more directly. In order to compare the automatically learned hierarchies with a reference hierarchy, we build on the work of [12] in which ontologies are compared along different levels: semiotic, syntactic and pragmatic. In particular, the authors present measures to compare the lexical and taxonomic overlap between two ontologies. In order to formally define our evaluation measures, we introduce a *core ontology* in line with [6] as follows:

Definition 1 (Core Ontology)
A core ontology is a structure $O := (C, \leq_C)$ consisting of (i) a set C of concept identifiers and (ii) partial order $\leq_C$ on C called concept hierarchy or taxonomy.

For the sake of notational simplicity we adopt the following convention: given an ontology O_i, the corresponding set of concepts will be denoted by C_i and the partial order representing the concept hierarchy by $\leq_{C_i}$.
It is important to mention that in the approach presented here, terms are directly identified with concepts, i.e. we neglect the fact that terms can be polysemous. In order to compare the taxonomy of two ontologies, we use the *semantic cotopy* (SC) presented in [12]. The semantic cotopy of a concept is defined as the set of all its super- and subconcepts:

$$SC(c_i, O_i) := \{c_j \in C_i \mid c_i \leq_C c_j \text{ or } c_j \leq_C c_i\},$$

Now, according to Maedche et al. the taxonomic overlap ($\overline{TO}$) of two ontologies O_1 and O_2 is computed as follows:

$$\overline{TO}(O_1, O_2) = \frac{1}{|C_1|} \sum_{c \in C_1} TO(c, O_1, O_2)$$

where

$$TO(c, O_1, O_2) := \begin{cases} TO'(c, O_1, O_2) & \text{if} c \in C_2 \\ TO''(c, O_1, O_2) & \text{if} c \notin C_2 \end{cases}$$

and TO' and TO" are defined as follows:

$$TO'(c, O_1, O_2) := \frac{|SC(c,O_1,O_2) \cap SC(c,O_2,O_1)|}{|SC(c,O_1,O_2) \cup SC(c,O_2,O_1)|}$$

$$TO''(c, O_1, O_2) := max_{c' \in C_2} \frac{|SC(c,O_1,O_2) \cap SC(c',O_2,O_1)|}{|SC(c,O_1,O_2) \cup SC(c',O_2,O_1)|}$$

So, TO' gives the similarity between concepts which are in both ontologies by comparing their respective semantic cotopies. In contrast, TO'' gives the similarity between a concept $c \in C_1$ and that concept c' in C_2 which maximizes the overlap of the respective semantic cotopies, i.e. it makes an optimistic estimation assuming an overlap that just does not happen to show up at the immediate lexical surface. The taxonomic overlap $\overline{TO}(O_1, O_2)$ between the two ontologies is then calculated by averaging over all the taxonomic overlaps of the concepts in C_1. To evaluate the automatically clustered concept hierarchies, we compare them with our gold standard by the taxonomic overlap measures described above. Given an automatically learned ontology O_{auto} and a reference ontology O_{ref}, calculating $\overline{TO}(O_{auto}, O_{ref})$ amounts to calculating the precision of O_{auto} with respect to O_{ref} as we calculate the taxonomic overlap for each concept in O_{auto}. In order to assess how satisfactory the coverage of the automatically learned ontology is with respect to the reference ontology, we need to compute also the inverse precision or recall, i.e. $Rec(O_{auto}, O_{ref}) = Prec(O_{ref}, O_{auto}) = \overline{TO}(O_{ref}, O_{auto})$. As we want to maximize both recall and precision, we evaluate our approach in terms of the following F-Measure:

$$F_{TO}(O_{auto}, O_{ref}) = \frac{2\ \overline{TO}(O_{auto}, O_{ref})\ \overline{TO}(O_{ref}, O_{auto})}{\overline{TO}(O_{auto}, O_{ref}) + \overline{TO}(O_{ref}, O_{auto})}$$

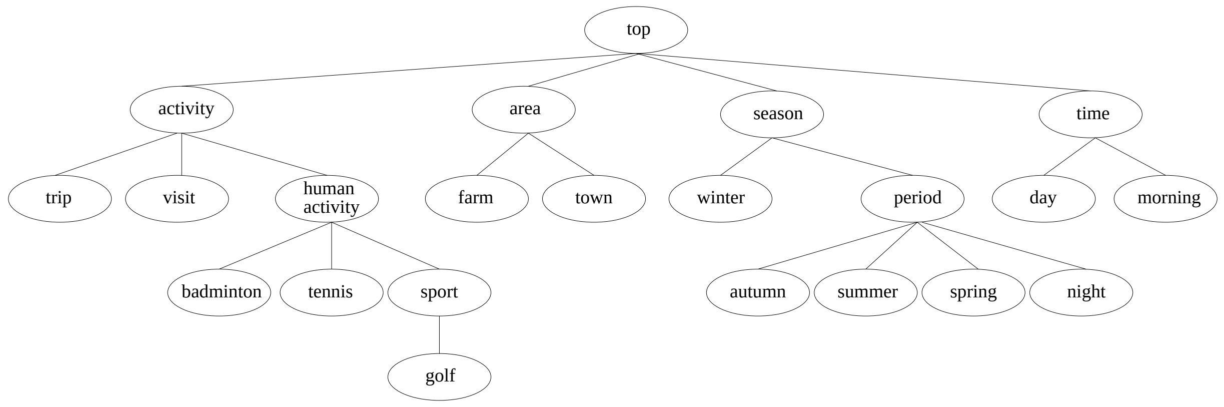

Figure 3. Example for an automatically learned concept hierarchy

To support our claim that our algorithm produces better groupings or clustering of terms, we also introduce the notion of *sibling overlap* (SO). For this purpose we define first what the siblings of a concept are, i.e. all its children:

$$Sib(c, O_i) := \{c' \mid c' \prec_{C_i} c\}$$

where $\prec_C$, the immediate predecessor relation is defined as follows:

Definition 2 ($\prec_C$)
$c' \prec_C c$ iff $c' \leq_C c$ and there is no c'' such that $c' \leq_C c''$ and $c'' \leq_C c$.

Finally, the average sibling overlap is defined as follows:

$$\overline{SO}(O_1, O_2) := \sum_{c_1 \in C_1} max_{c_2 \in C_2} \frac{|Sib(c_1,O_1) \cap Sib(c_2,O_2)|}{|Sib(c_1,O_1) \cup Sib(c_2,O_2)|}$$

Here we also calculate the F-Measure as follows:

$$F_{SO}(O_{auto}, O_{ref}) = \frac{2\ \overline{SO}(O_{auto}, O_{ref})\ \overline{SO}(O_{ref}, O_{auto})}{\overline{SO}(O_{auto}, O_{ref}) + \overline{SO}(O_{ref}, O_{auto})}$$

3.1. Results

As text collection we use texts acquired from *http://www.lonelyplanet.com* as well as from *http://www.all-in-all.de*. Furthermore, we also use a general corpus, the British National Corpus. Altogether the corpus size was over 118 Million tokens. The reference ontology is the one of the comparison study in [12], which was modeled by an experienced ontology engineer. The tourism domain ontology consists of 293 concepts and can be downloaded at http://www.aifb.uni-karlsruhe.de/WBS/pci/TourismGoldStandard.isa.

3.2. Comparison with a Reference Taxonomy

Table 2 shows the results of comparing the concept hierarchies produced by our method with the reference concept hierarchy in terms of the taxonomic overlap measures. In particular, the table shows results for different combinations of the resources used for the construction of the oracle. The best results of F$_{TO}$=23.11% were achieved on the one hand using a combination of WordNet and Hearst Patterns as well as a combination of Hearst patterns matched in the corpus and on the WWW on the other hand. Of all the resources considered, WordNet and the patterns matched in the WWW yield the worst results when used alone or in combination. Table 3 shows the results in terms of sibling overlap. The best result of F$_{SO}$=14.18% is achieved when using all the three resources for the hypernym extraction. It is interesting to observe that with respect to sibling overlap, the most reliable source for the hypernym extraction are the patterns matched on the WWW.

3.3. Comparison with Caraballo's Method

In order to evaluate our approach we implemented the method described in [2] in which first a hierarchy is produced by standard agglomerative clustering and then hypernyms derived from Hearst patterns are attached to each cluster. The most frequent hypernym is then taken as a label for the cluster provided that it is a valid hypernym for at least two elements in the cluster. Finally, the hierarchy is compressed by removing all clusters without a label. In our implementation of this method we used *complete linkage* as strategy to calculate the similarity between clusters (compare [3]) and used our hypernym oracle instead of merely the hypernyms derived from Hearst patterns. Table 4 shows the results of Caraballo's method in terms of taxonomic overlap with the reference standard. The best result of F_{TO}=22.48% is achieved using only WordNet as resource for the hypernym extraction. The results are however worse compared to our guided agglomerative clustering algorithm. Even more decisive are the results in terms of sibling overlap reported in Table 5, showing that our method clearly outperforms Caraballo's approach in terms

	$\overline{TO}$(auto,ref)	$\overline{TO}$(ref,auto)	F$_{TO}$(auto,ref)
WordNet + Hearst + WWW	18.62%	18.63%	18.62%
WordNet + Hearst	23.05%	23.16%	**23.11%**
WordNet + WWW	18.27%	18.04%	18.15%
Hearst + WWW	23.05%	23.16%	**23.11%**
WordNet	19.18%	19.18%	19.18%
Hearst	22.15%	22.09%	22.12%
WWW	19.17%	19.06%	19.12%

Table 2. Results for Guided Agglomerative Clustering in terms of $\overline{TO}$

	$\overline{SO}$(auto,ref)	$\overline{SO}$(ref,auto)	F$_{SO}$(auto,ref)
WordNet + Hearst + WWW	12.99%	15.61%	**14.18%**
WordNet + Hearst	13.03%	12.20%	12.60%
WordNet + WWW	13.31%	14.90%	14.06%
Hearst + WWW	13.21%	15.02%	14.06%
WordNet	13.27%	11.64%	12.40%
Hearst	12.78%	12.30%	12.54%
WWW	12.75%	13.94%	13.32%

Table 3. Results for Guided Agglomerative Clustering in terms of $\overline{SO}$

of cluster coherence. In fact, the best result (F_{SO}=14.18%) of the guided agglomerative algorithm is more than 5 points above the best result achieved with Caraballo's method, i.e. $F_{SO} = 8.96\%$.

3.4. Human Assessment

As Sabou et al. [15] have shown, using a gold standard for the evaluation of automatically constructed ontologies is sometimes problematic and may lead to wrong conclusions about the quality of the learned ontology. This is due to the fact that if the learned ontology does not mirror the gold standard, it does not necessarily mean that it is wrong. In order to assess the quality of the automatically learned concept hierarchies more directly, we thus asked a student at our institute to validate the learned isa-relations by assigning credits from 3 (correct), over 2 (almost correct) and 1 (not completely wrong) to 0 (wrong). Actually, we did not consider those *isa*-relations classifying a concept directly under root as it seems very difficult to assess what should be directly under root and what not. Then we calculated the precision of the system counting an isa-relation as correct if it received three credits (P_3), at least two credits (P_2) and at least one credit (P_1), respectively. The precision for the versions of our approach using different combinations of the hypernym resources are given in Table 6, showing also the number of isa-relations evaluated. The results show on the one hand that the concept hierarchies produced by our method are quite reasonable according to human intuitions. Actually, the fact that 65.66% of the learned relations are considered as totally correct by our evaluator is a very impressive result. A second interesting conclusion is that the version of our algorithm combining all the different resources for the oracle construction performed better compared the ones using any subset of them.

4. Comparison and Related Work

Many approaches to learning conceptual hierarchies exploit Harris' distributional hypothesis and cluster terms on the basis of their contextual similarity with respect to a given corpus. Most work based on this hypothesis relies on agglomerative hierarchical clustering algorithms such as [7], [2] and [1]. Others use soft clustering algorithms such as the one presented in [13] which uses deterministic annealing to find lowest distortion sets of clusters. A partitional algorithm, viz. Bi-Section-KMeans, as well as a set-theoretic approach, i.e. Formal Concept Analysis, are for example used in [3]. Two applications of non-hierarchical clustering algorithms to learning clusters of terms are described in [14] and [11], respectively. Common to these clustering approaches is the problem that high degrees of similarity can in fact be accidental and actually due to the corpus used, which typically represents only a biased and very small portion of the actual world or of a certain domain. By guiding the clustering process by an external hypernym oracle, the approach presented in this paper is able to reduce such accidental clusterings and thus increase the quality of the learned concept hierarchies.
Another drawback common to the above mentioned clustering approaches is the lack of appropriate labels for clusters. Though appropriate labels are strictly not necessary,

	$\overline{TO}$(auto,ref)	$\overline{TO}$(ref,auto)	F$_{TO}$(auto,ref)
WordNet + Hearst + WWW	14.88%	16.13%	15.48%
WordNet + Hearst	20.09%	20.82%	20.45%
WordNet + WWW	15.06%	16.32%	15.67%
Hearst + WWW	14.87%	16.00%	15.42%
WordNet	22.22%	22.77%	**22.48%**
Hearst	19.86%	20.58%	20.21%
WWW	15.05%	16.19%	15.60%

Table 4. Results for Caraballo's method in terms of $\overline{TO}$

	$\overline{SO}$(auto,ref)	$\overline{SO}$(ref,auto)	F$_{SO}$(auto,ref)
WordNet + Hearst + WWW	7.67%	10.72%	8.94%
WordNet + Hearst	9.36%	6.09%	7.38%
WordNet + WWW	7.75%	10.70%	8.90%
Hearst + WWW	7.68%	10.64%	8.92%
WordNet	7.68%	1.26%	2.16%
Hearst	9.33%	5.90%	7.23%
WWW	7.7%	10.62%	**8.96%**

Table 5. Results for Caraballo's method in terms of $\overline{SO}$

they allow a better readability of the learned hierarchies for humans. On the other hand the lack of labels makes also the evaluation of the learned structures more difficult when comparing to a certain gold standard, requiring a notion of similarity such as proposed in [12]. Caraballo [2] addresses the labeling problem and after producing an unlabeled cluster tree, she also labels the abstract concepts of the hierarchy by considering the Hearst patterns in which the children of the concept in question appear as hyponyms. The most frequent hypernym is then chosen in order to label the concept. Though our approach is similar, it also crucially differs in the fact that in Caraballo's approach the clustering process is independent of the labeling, while in our approach they are integrated with each other, producing overall better hierarchies. In the approach of [3] using Formal Concept Analysis, the labeling problem is tackled by naming a concept with the intent of the corresponding formal concept, containing verb-derived attributes automatically extracted from the corpus. However, the quality of this labeling procedure is not evaluated. On the other hand, intents in Formal Concept Analysis can get very large so that in many cases such an approach will produce too large labels.

5. Conclusion

We have presented a novel guided agglomerative clustering algorithm with the aim of automatically inducing concept hierarchies from a text corpus. The algorithm exploits an external hypernym oracle to drive the clustering process. Further, we have also described an automatic method to derive such a hypernym oracle from WordNet, a corpus as well as the WWW. The approach has been evaluated by comparing the resulting concept hierarchies with a reference concept hierarchy for the tourism domain. In fact, we have shown that the results of our algorithm are better when compared to Caraballo's approach. The human assessment of the automatically produced concept hierarchy has also shown that the learned relations are reasonably precise. Besides overcoming two main problems of unsupervised approaches, i.e. accidental clusterings as well as lack of labels, our approach also is original in that it successfully combines two main paradigms to ontology learning: the approaches relying on contextual similarity as well as approaches matching lexico-syntactic patterns denoting a certain relation such as in [10].

Acknowledgments The work reported in this paper has been partially supported by the SmartWeb project[4], funded by the German Ministry of Research. Thanks also to Laura Goebes for assisting in the evaluation of the system.

References

[1] G. Bisson, C. Nedellec, and L. Canamero, 'Designing clustering methods for ontology building - The Mo'K workbench', in *Proceedings of the ECAI Ontology Learning Workshop*, pp. 13–19, (2000).

[2] S.A. Caraballo, 'Automatic construction of a hypernym-labeled noun hierarchy from text', in

[4]http://www.smartweb-projekt.de/

	#	P_1	P_2	P_3
WordNet + Hearst + WWW	265	67.17%	**66.04%**	**65.66%**
WordNet + Hearst	233	65.24%	62.23%	62.23%
WordNet + WWW	262	68.32%	65.65%	65.65%
Hearst + WWW	268	**69.03%**	63.43%	63.43%
WordNet	236	58.90%	55.51%	55.08%
Hearst	203	66.50%	64.04%	64.04%
WWW	261	73.18%	64.37%	62.07%

Table 6. Results of the human evaluation of the hierarchies produced by our guided clustering algorithm

Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, pp. 120–126, (1999).

[3] P. Cimiano, A. Hotho, and S. Staab, ‘Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text’, in *Proceedings of the European Conference on Artificial Intelligence*, pp. 435–439, (2004).

[4] P. Cimiano, L. Schmidt-Thieme, A. Pivk, and S. Staab, ‘Learning taxonomic relations from heterogeneous evidence’, in *Ontology Learning from Text: Methods, Applications and Evaluation*, eds., P. Buitelaar, P. Cimiano, and B. Magnini, IOS Press, (2005). to appear.

[5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons, Inc., 2001.

[6] G. Stumme et al., ‘The karlsruhe view on ontologies’, Technical report, University of Karlsruhe, Institute AIFB, (2003).

[7] D. Faure and C. Nedellec, ‘A corpus-based conceptual clustering method for verb frames and ontology’, in *Proceedings of the LREC Workshop on Adapting lexical and corpus resources to sublanguages and applications*, ed., P. Velardi, pp. 5–12, (1998).

[8] C. Fellbaum, *WordNet, an electronic lexical database*, MIT Press, 1998.

[9] Z. Harris, *Mathematical Structures of Language*, Wiley, 1968.

[10] M.A. Hearst, ‘Automatic acquisition of hyponyms from large text corpora’, in *Proceedings of the 14th International Conference on Computational Linguistics*, pp. 539–545, (1992).

[11] D. Hindle, ‘Noun classification from predicate-argument structures’, in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 268–275, (1990).

[12] A. Maedche and S. Staab, ‘Measuring similarity between ontologies’, in *Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW)*, pp. 251–263. Springer Verlag, (2002).

[13] F. Pereira, N. Tishby, and L. Lee, ‘Distributional clustering of english words’, in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 183–190, (1993).

[14] M-L Reinberger and W. Daelemans, ‘Unsupervised text mining for ontology extraction: an evaluation of statistical measures’, in *Proceedings of the International Conference on Lexical Resources and Evaluation (LREC)*, pp. 491–494, (2004).

[15] Marta Sabou, ‘Learning web service ontologies: an automatic extraction method and its evaluation’, in *Ontology Learning from Text: Methods, Applications and Evaluation*, eds., P. Buitelaar, P. Cimiano, and B. Magnini. IOS Press, (2005). to appear.

[16] H. Schmid, ‘Probabilistic part-of-speech tagging using decision trees’, in *Proceedings of the International Conference on New Methods in Language Processing*, (1994).

[17] P. Velardi, P. Fabriani, and M. Missikoff, ‘Using text processing techniques to automatically enrich a domain ontology’, in *Proceedings of the ACM International Conference on Formal Ontology in Information Systems*, (2001).

[18] G. Zipf, *Selective Studies and the Principle of Relative Frequency in Language*, Cambridge, 1932.