Comparing Conceptual, Divisive and Agglomerative Clustering for Learning Taxonomies from Text

Philipp Cimiano[1], **Andreas Hotho**[1] **and Steffen Staab**[1]

Abstract. The application of clustering methods for automatic taxonomy construction from text requires knowledge about the tradeoff between, *(i)*, their effectiveness (quality of result), *(ii)*, efficiency (run-time behaviour), and, *(iii)*, traceability of the taxonomy construction by the ontology engineer. In this line, we present an original conceptual clustering method based on Formal Concept Analysis for automatic taxonomy construction and compare it with hierarchical agglomerative clustering and hierarchical divisive clustering.

1 Introduction

In order to reduce efforts for engineering large ontologies, recent years have seen a surge of interests for learning ontologies from text in general and learning of taxonomies, i.e. concept hierarchies, in particular. The principle paradigm exploited in many of these approaches is to derive knowledge from texts by analyzing how certain terms are used. The *distributional hypothesis* [9] assumes that terms are similar to the extent to which they share similar linguistic contexts and thus gives rise to various methods that cluster terms based on their linguistic context and form corresponding taxonomies.

In spite of increased interest in ontology learning and in spite of a growing number of approaches that have been investigated, we see two drawbacks currently prevailing in the field:

1. Ontology engineers who are interested in applying such methods still lack an adequate comparison in order to choose between available methods.
2. The set of unsupervised methods for automatic taxonomy construction has so far been very focused on agglomerative clustering methods, while the more efficient partitional clustering methods and conceptual clustering, which leads to more easily traceable taxonomy construction, have been investigated only to a very small extent and not at all, respectively.

In this paper we examine different clustering paradigms that exploit the distributional hypothesis to automatically construct taxonomies. We provide an original conceptual clustering method based on Formal Concept Analysis for automatic taxonomy construction and we compare it against standard hierarchical agglomerative clustering and Bi-Section KMeans as an instance of a divisive algorithm.

We discuss the approaches comparing them along the dimensions of, *(i)*, effectiveness (i.e. quality of result), *(ii)*, efficiency (i.e. runtime behaviour) and, *(iii)*, traceability of taxonomy construction by the ontology engineer. We base the discussion on, *(i)*, a manually built gold standard, *(ii)*, results from the literature, and, *(iii)*, a qualitative discussion of how easy it is for the ontology engineer to comprehend why the taxonomy was constructed in a particular way by the corresponding method.

The reminder of this paper is organized as follows: Section 2 describes the text processing methods used to describe terms by their linguistic context. Section 3 introduces the different clustering paradigms, and more concretely the approaches, we consider in our comparison. Then, Section 4 compares the different approaches. Before concluding, we discuss some related work in Section 5.

2 Text Processing

We describe the linguistic context of a term by the syntactic dependencies that it establishes as the head of a subject, of an object or of a PP-complement with a verb. Then, we represent a term by its context, i.e. by a vector, the entries of which count the frequency of syntactically dominating verbs. In order to determine the frequencies of such dependencies automatically, we parse the text with LoPar, a trainable, statistical left-corner parser[2]. From the parse trees we then extract the syntactic dependencies between a verb and its subject, object and PP complement by using tgrep[3]. Finally, we also lemmatize the verbs as well as the head of the subject, object and PP complement by looking up the lemma in the lexicon provided with LoPar[4]. Let's take for instance the following two (made-up) sentences:

People book hotels. The man drove the bike along the beach.

After parsing these sentences, we would extract the following syntactic dependencies (left) and after lemmatizing we would get the pairs on the right:

book_subj(people)	book_subj(people)
book_obj(hotels)	book_obj(hotel)
drove_subj(man)	drive_subj(man)
drove_obj(bike)	drive_obj(bike)
drove_along(beach)	drive_along(beach)

Furthermore, as (i) the output of the parser can be erroneous, i.e. not all derived verb/object dependencies are correct, and (ii) not all the derived dependencies are 'interesting' in the sense that they will help to discriminate between the different objects, we weigh the verb/object dependencies with regard to a certain information measure. Thus, we consider only those verb/object relations for which this information measure is above some threshold t. In earlier work

[1] AIFB, University of Karlsruhe, Germany;{cimiano,staab,hotho}@aifb.uni-karlsruhe.de

[2] http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/LoPar-en.html

[3] see http://mccawley.cogsci.uiuc.edu/corpora/treebank3.html

[4] The main difference to our earlier work presented in [3] is that here we performed lemmatization and also improved the accuracy of the syntactic dependency process this yielding better results.

we experimented with different information measures and found out that the conditional probability works well enough ([3]). Thus, we calculate the conditional probability that a certain (multi-word) term n appears as head of a certain argument position arg of a verb v as follows: $P(n|v_{arg}) = \frac{f(n,v_{arg})}{f(v_{arg})}$, where $f(n, v_{arg})$ is the number of occurrences of a term n in the argument position arg of a verb v and $f(v_{arg})$ is the number of occurrences of argument position arg for a verb v.

3 Clustering Approaches

Different methods have been proposed in the literature to address the problem of (semi-) automatically deriving a concept hierarchy from text via clustering. They can be grouped in two classes: the *similarity*-based methods on the one hand and the *set-theoretical* approaches on the other hand. Both methods adopt a vector-space model and represent a word or term as a vector containing features or attributes derived from a certain corpus.

The first type of methods is characterized by the use of a similarity/distance measure in order to compute the pairwise similarity/distance between vectors corresponding to terms in order to decide if they are semantically similar and thus should be clustered or not. In our experiments, for example, we measure the similarity between terms by the cosine of the angle between the vectors $\vec{t_1}, \vec{t_2}$ representing them, i.e. $\cos(\sphericalangle(\vec{t_1}, \vec{t_2})) = \frac{\vec{t_1} \cdot \vec{t_2}}{\|\vec{t_1}\| \cdot \|\vec{t_2}\|}$
The similarity-based clustering algorithms are further categorized into *agglomerative* (bottom-up) and *divisive* (top-down). Some prominent examples for this type of method are [2, 10, 6, 12, 1].

In contrast, set-theoretic approaches partially order the objects according to the inclusion relationship between their feature sets ([8]).

In what follows we first describe the different clustering approaches we used in our experiments, i.e. a set-theoretic approach based on FCA and two similarity-based approaches, viz. a hierarchical agglomerative clustering algorithm and Bi-Section-KMeans as an instance of a divisive algorithm.

3.1 Formal Concept Analysis

Formal Concept Analysis (FCA) uses order theory to analyze the correlations between objects, G, and their features, M. FCA identifies from such a data description, a so called *formal context* $\mathbb{K}$, its set of features $B \subseteq M$ being bijectively correlated with its set of objects $A \subseteq G$. Such a correlated pair is called a *formal concept* (A, B). [5] Formal concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2$ or, which is equivalent, $B_2 \subseteq B_1$. Objects $o_1, o_2 \in G$ are conceptually clustered iff $\{o_1, o_2\} \subseteq A$, where (A, B) is a formal concept in $\mathbb{K}$. We illustrate FCA with a brief example. From a corpus of text on tourism, we may derive features as described in section 2 and construct a vector representation for objects G={ *apartment*, *car*, *bike*, *trip*, *excursion* }. The representation with features $M := \{\text{bookable}, \ldots, \text{joinable}\}$ is given in table 1. It describes that all given objects can be booked. Furthermore, one can rent a *car*, a *bike* or an *apartment*. Moreover, one can drive a *car* or a *bike*, but one may only ride a *bike*. Eventually, it is derived from the corpus that one may join an *excursion* or a *trip*.

From this binary representation of objects, FCA constructs a lattice of formal concepts. The lattice corresponding to our running example is depicted in figure 1 (left).[6] In order to interpret it as a taxonomy, we apply the following two rules:

	bookable	rentable	driveable	rideable	joinable
apartment	x	x			
car	x	x	x		
motor-bike	x	x	x	x	
excursion	x				x
trip	x				x

Table 1. Tourism domain knowledge as formal context

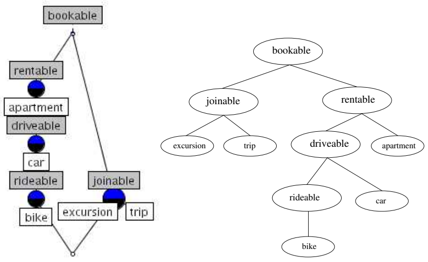

Figure 1. The lattice and concept hierarchy for the tourism example

1. Introduce one *taxonomy concept* c_B *labeled with* B *for each* formal concept (A, B) iff $|A| \geq 2$. Order them according to the order in the lattice.
2. Introduce one *taxonomy concept* c_o *for each object* $o \in G$ *and label it by o. Order them such that* $c_o \leq c_B$ *where* (A, B) *is a formal concept,* $o \in A$ *and there is no formal concept* (A', B') *such that* $(A', B') \leq (A, B)$ *and* $o \in A'$.

Traceability By our FCA-based method we derive a taxonomy mostly with (object) terms and partially with verb-like identifiers. This view of the taxonomy is quite natural when interpreting it in terms of logical subsumption, e. g.: $\forall x\ (bike(x) \rightarrow rideable(x))$ We observe that the 'verb-like' concept identifiers have the same status as any other concept identifier from an extensional point of view. Even going further, in some cases, there may not even exist an appropriate hypernym in the language to label a certain abstract concept, such that using a verb-like identifier might even be necessary. For example, we could easily replace the identifiers *joinable*, *rideable*, *driveable* by *activity*, *two-wheeled vehicle* and *vehicle*, respectively. However, it is certainly difficult to replace *rentable* by some meaningful term denoting the same extension, i. e. all the things that can be rented. Thus, FCA is a method that allows very well for tracing the reasons that a taxonomy has been constructed the way it is.

Efficiency Finally, we notice from the literature [8] that the runtime of FCA is exponential in the minimum of the number of objects and the number of features. Thus, its general time complexity is $O(2^n)$ in the worst case, where n is the number of terms to be ordered, as the set of features typically outnumbers the set of terms to be ordered.

3.2 Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering (compare [5]) is a similarity-based bottom-up clustering technique in which at the beginning every term forms a cluster of its own. Then the algorithm iterates over the step that merges the two most similar clusters still available, until one arrives at a universal cluster that contains all the terms.
In our experiments, we use three different strategies to calculate the

[5] The reader is referred to [8] for the formal definition of *formal context*, *formal concept*, and the subconcept-superconcept relation between formal concepts.

[6] The Concept Explorer software was used to produce this visualization (see *http://sourceforge.net/projects/conexp*).

similarity between clusters: *complete-*, *average-* and *single*-linkage. The three strategies may be based on the same similarity measure between terms, i.e. the cosine measure in our experiments, but they measure the similarity between two non-trivial clusters in different ways.

Single linkage defines the similarity between two clusters P and Q to equal $\max_{p\in P,q\in Q} sim(p,q)$, considering the closest pair between the two clusters. *Complete* linkage considers the two most dissimilar terms, i.e. $\min_{p\in P,q\in Q} sim(p,q)$. Finally, *average-linkage* computes the average similarity of the terms of the two clusters, i.e. $\frac{1}{|P||Q|}\sum_{p\in P,q\in Q} sim(p,q)$. The reader should note that we prohibit the merging of clusters with similarity 0 and rather order them under a fictive universal cluster 'root'.

Traceability. Similarity measures in a high-dimensional space — like the vector representations we consider here — and their consequences in agglomerative clustering are typically hard to trace by the ontology engineer. Insight into the constructed taxonomy is usually restricted to understanding that a few initial merges of individual terms into small-sized clusters are meaningful given the concrete text corpus.

Efficiency. The time complexity of naive implementations of hierarchical agglomerative clustering algorithms is $O(n^3)$ where n is the number of terms. Optimized implementations achieve $O(n^2 \log n)$ (cf. [4]). The time complexity when using single-linkage as linkage metric is even $O(n^2)$.

3.3 Bi-Section-KMeans

In [13] it has been shown that Bi-Section-KMeans – a variant of KMeans – is a good and fast divisive clustering algorithm. It frequently outperforms standard KMeans as well as agglomerative clustering techniques.

Bi-Section-KMeans is defined as an outer loop around standard KMeans. In order to generate k clusters, Bi-Section-KMeans repeatedly applies KMeans. Bi-Section-KMeans is initiated with the universal cluster containing all terms. Then it loops: It selects the cluster with the largest variance[7] and it calls KMeans in order to split this cluster into exactly two subclusters. The loop is repeated $k-1$ times such that k non-overlapping subclusters are generated. As similarity measure we also use the cosine measure as defined above. Further, as Bi-Section-KMeans is a randomized algorithm, we produce ten runs and average the obtained results.

Traceability. Concerning traceability, Bi-Section-KMeans shares the problem that similarities in high-dimensional space are difficult to understand. In contrast to agglomerative algorithms, Bi-Section-KMeans may incur that the two most similar terms are still split into different clusters, as a wrong decision at the upper level of generalization may jeopardize intuitive clusterings at the lower level.

Efficiency. The time complexity of Bi-Section-KMeans algorithms is $O(nk)$ where n is the number of terms and k is the number of clusters. In our setting we are interested in the complete tree for all terms produced by Bi-Section-KMeans. Thus, k is equal to n and the overall complexity is $O(n^2)$.

4 Evaluation

In order to evaluate the different clustering approaches, we compare the automatically generated concept hierarchies against a handcrafted ontology for the tourism domain. Actually, we used the reference ontology of the comparison study in [11], which was modeled in German by an experienced ontology engineer. Further, we manually added the English labels for those concepts whose German label has an English counterpart with the result that most of the concepts (>95%) finally came with an English label, too.[8] The resulting tourism domain ontology comprised 289 concepts.

The task we are evaluating against is as follows, given the 289 concepts of this ontology, our aim is to reproduce the hierarchy by applying clustering techniques. In this line we are evaluating how similar the automatically produced hierarchies and the reference hierarchy are.

One of the few approaches to compare the similarity of two ontologies as a whole can be found in [11]. There ontologies are compared at a lexical as well as at a semantic level. Following their approach, we present a comparison based on lexical overlap as well as taxonomic similarity between ontologies.

The core ontological model on which we base our evaluation is defined as follows:

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

As we injectively map terms onto concepts in all three clustering approaches, we neglect the fact that terms can be polysemous.[9] Now, the Lexical Overlap (LO) and Lexical Recall (LR) of two ontologies O_1 and O_2 are measured as follows:

$$LO(O_1,O_2) = \frac{|C_{O_1} \cap C_{O_2}|}{|C_{O_1} \cup C_{O_2}|} \text{ and } LR(O_1,O_2) = \frac{|C_{O_1} \cap C_{O_2}|}{|C_{O_2}|}$$

Further, we also introduce the lexical recall LR' which measures the recall of the ontology O_1 against those terms/concepts in O_2 which appear as arguments of a certain verb in the results of our syntactic dependency extraction process which results in the set C'_{O_2}:

$$LR'(O_1,O_2) = \frac{|C_{O_1} \cap C'_{O_2}|}{|C'_{O_2}|}$$

This variant of the lexical recall was introduced in order not to penalize the system for not producing information for terms which are not in our dataset.

In order to compare the taxonomy of the ontologies, we use the *semantic cotopy* (SC) presented in [11]. The semantic cotopy of a concept is defined as the set of all its super- and subconcepts:

$$SC(c_i, O) := \{c_j | c_j \in C_O \wedge (c_i \leq_C c_j \vee c_j \leq_C c_i)\},$$

where $c_i \in C_O$. For example, the semantic cotopy of *bike* in the concept hierarchy in Figure 1 (right) would be {*rideable, driveable, rentable, bookable*}. However, when comparing the similarity between an automatically clustered ontology and a human-modeled one, in order to have a fair comparison we should exclude (i) concepts such as *rideable, driveable, rentable* etc. from the automatically created hierarchy and (ii) abstract concepts such as *partially_material_thing* from the reference standard when computing the semantic cotopy. Thus, given two ontologies O_1 and O_2 we will only consider the common concepts in the semantic

[7] Though we don't make use of it in our experiments, it is also possible to select the largest cluster for splitting.

[8] Some concepts did not have a direct counterpart in English.

[9] In principle, FCA is able to account for polysemy of terms; however, we will gloss over this aspect in the present paper.

cotopy SC', i.e.

$$SC'(c_i, O_1, O_2) := \{c_j | c_j \in C_1 \cap C_2 \wedge (c_j \leq_{C_1} c_i \vee c_i \leq_{C_1} c_j)\}.$$

Now the taxonomic overlap of two ontologies O_1 and O_2 is computed as follows according to [11]:

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

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

and TO' and TO" are defined as follows (compare [11]):

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

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

In order to balance the lexical recall and the taxonomic overlap against each other, we compute the F-Measure of them as follows: $F(A, B) = \frac{2*A*B}{A+B}$.
In particular, we evaluate an automatically produced ontology O_{AUTO} by evaluating how much of the terms in O_{REF} it is able to order hierarchically, i.e. calculating $LR(O_{AUTO}, O_{REF})$ or $LO(O_{AUTO}, O_{REF})$ as well as how much of the taxonomy of O_{REF} it is able to reproduce, i.e. calculating $\overline{TO}(O_{REF}, O_{AUTO})$ Finally we balance these two values by the F-Measure described above. Thus, henceforth LR, LR', LO and $\overline{TO}$ will stand proxy for $LR(O_{AUTO}, O_{REF})$, $LR'(O_{AUTO}, O_{REF})$, $LO(O_{AUTO}, O_{REF})$ and $\overline{TO}(O_{REF}, O_{AUTO})$, respectively.

4.1 Results

As domain-specific text collection we use texts acquired from *http://www.lonelyplanet.com* as well as from *http://www.all-in-all.de*. Furthermore, we also used a general corpus, the British National Corpus. Figure 2 (left) shows the values of the lexical measures LO, LR as well as LR' for the results of the FCA-based approach. Obviously, the lexical recall measures LR and LR' decrease with higher thresholds t for the conditional probability $P(n|v_{arg})$ as the more information we cut off, the less lexical material we will have. In contrast, we can also observe that the lexical overlap (LO) increases proportionally to the threshold t. The reason for this is that the higher the threshold, the smaller the automatically produced ontologies get.[10] Overall, the best results for the measures are a LR of 44.56% (t=0.005), a LR' of 79.40%[11] (t=0.005) and a LO of 37.63% (t=0.7). Figure 2 (left) also shows the results for the taxonomic overlap ($\overline{TO}$) calculated by using the modified semantic cotopy SC' which takes into account only concepts common to both ontologies. Figure 2 (right) shows the F-Measures for the different combinations. The measures are ordered bottom-up according to the 'fairness' with the system. The best result of the base F-Measure $F(LO, \overline{TO})$ is 43.83% (t=0.7), for $F(LR, \overline{TO})$ we get 51.10% (t=0.005) and the best result of $F(LR', \overline{TO})$ is an impressive 68.23% (t=0.005).

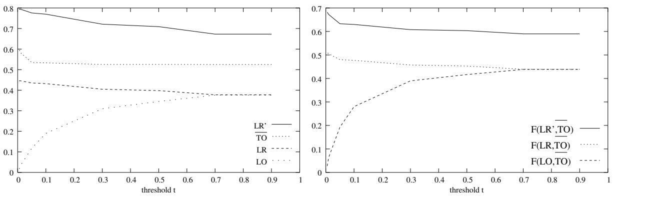

Figure 2. Results of the lexical, taxonomic and F-measures over threshold t (FCA-based approach)

Compared to the agglomerative hierarchical clustering algorithm as well as to Bi-Section-KMeans, we get the results depicted in Figure 3[12]. The best result is achieved by the FCA-based approach with the above mentioned $F(LR', \overline{TO})$ = 68.23%, followed by the agglomerative clustering approach with $F(LR', \overline{TO})$ = 62.92% (*complete-linkage*), $F(LR', \overline{TO})$ = 62.84% (*average-linkage*) and $F(LR', \overline{TO})$ = 62.78% (*single-linkage*). Bi-Section-KMeans achieves a best result of $F(LR', \overline{TO})$ = 62.80%. It is important to mention that from threshold 0.7 on, the elements have no common features anymore such that the produced hierarchies are either flat with all terms directly under the root node as in the FCA-based and agglomerative approaches or just a binary tree produced by random splits in the case of Bi-Section-Kmeans. The result here is $F(LR', \overline{TO})$ = 58.97% , which constitutes the baseline to compare with.

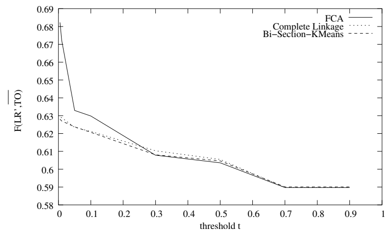

Figure 3. Comparison: results of $F(LR', \overline{TO})$ over threshold t for the FCA-based approach, hierarchical agglomerative clustering with complete linkage and Bi-Section-KMeans

Discussion. In order to compare the performance of our approach with the human performance on the task, we interpret our results with regard to the study presented in [11]. In this study, four subjects were asked to model a taxonomy on the basis of 310 lexical entries relevant for the tourism domain. The taxonomic overlap ($\overline{TO}$) between the manually engineered ontologies reached from 47% to 87% with an average of 56.35%. Thus, it is clear that any automatic approach to derive a conceptual hierarchy between a set of concepts has definitely its limits. Assuming a LR of 100% for humans on the task of modelling a concept hierarchy, i.e. being able to order all the terms in question, we thus get $F_{human}(LR, \overline{TO})$=72.1%, from which – even artificially creating a human-like recall by measuring $F(LR', \overline{TO})$ – all the approaches fall short of. However, in order to be fully comparable, we should perform an additional experiment in which humans hierarchically order the terms appearing in our dataset.
On the other hand, we also conclude that the problem of automatically constructing hierarchies from text by clustering techniques is inherently hard, which can be seen by the fact that the best approach is just over 9% above the baseline strategy corresponding to putting all terms under the root node.

Comparison. We have shown that the different clustering approaches we have compared exhibit similar effectiveness when automatically creating a taxonomy from text. The novel approach based on FCA even seems to achieve slightly better results than the other clustering approaches. In general, all the approaches fall short of human achievement, however they appear to be good enough to support

[10] The size of the automatically acquired hierarchy at $t = 0.005$ was of 7.236 concepts, which explains why LO is almost 0 at this threshold.
[11] The reason why the maximum lexical recall LR' is not 100% is because we started with a threshold of $t = 0.005$; when using no threshold, the lexical recall LR' is 100%

[12] Here we have not shown the results for *average-* and *single* linkage as they are very similar to the ones for *complete* linkage.

the human ontology engineer.

Furthermore, Table 2 summarizes previous sections and highlights the fact that every approach has its own benefits and drawbacks. The main benefit of using FCA is that it does not only generate clusters - formal concepts to be more specific - but it also provides an intensional description for these clusters thus contributing to better understanding by the ontology engineer (compare Figure 1 (left)). This is in contrast to the similarity-based methods, which do not provide the same level of traceability due to the fact that it is the numerical value of the similarity between two high-dimensional vectors which drives the clustering process and which thus remains opaque to the engineer. The agglomerative and divisive approach are different in this respect in that in the first, initial merges of small-size clusters correspond to high degrees of similarity and are thus more understandable, while in the latter the splitting of clusters aims at minimizing the overall cluster variance thus being harder to trace.

A clear disadvantage of FCA is that the size of the lattice can get exponential in the size of the context in the worst case thus resulting in an exponential time complexity — compared to $O(n^2 \log n)$ and $O(n^2)$ for agglomerative clustering and Bi-Section-KMeans, respectively. However, in our experiments FCA showed a near linear time complexity which is due to the fact that vector representations of terms are extremely sparse.

	Effectiveness	Efficiency	Traceability
FCA	Good	$O(2^n)$ Near Linear (empirical)	Good
Agglomerative Clustering	Good	$O(n^2 \log n)$ (complete/average) $O(n^2)$ (single)	Fair
Bi-Section-KMeans	Good	$O(n^2)$	Weak-Fair

Table 2. Trade-offs between different taxonomy construction methods

5 Related Work

Without doubt, there is a lack of evaluation standards and thus also of comparative work in the field of learning concept hierarchies from text by clustering. In the following, we mention some related approaches and briefly review their evaluation methodology.

There exist several approaches which are based on the distributional hypothesis and which make use of clustering techniques to derive term hierarchies from text by using certain syntactic dependencies. Faure et al. ([6]) present an iterative bottom-up clustering approach of nouns appearing in similar contexts. At each step, they cluster together the two most similar extents of some argument position of two verbs. Their approach is not unsupervised as the user has to validate the built clusters after each iteration. As results, they give the number of 'relevant' clusters produced (in terms of accuracy) in dependence of the percentage of the corpus used. Pereira et al. ([12]) present a top-down clustering approach to build an unlabeled hierarchy of nouns. As in our approach, they also make use of verb-object relations to represent the context of a certain noun. They evaluate their results on the one hand by considering the entropy of the produced clusters and also in an indirect way in the context of a linguistic decision task. Caraballo ([2]) also uses an agglomerative technique to derive an unlabeled hierarchy of nouns by using data on conjunctions of nouns and appositive constructs. She evaluates her approach by presenting the hypernyms and the hyponym candidates to users for validation. Another interesting approach is the incremental conceptual clustering presented in [7] which is based on a *category utility* as a quality measure to be maximized. Fisher evaluates his system in the context of a diagnosis prediction task in the medical domain.

It is clear that none of the above presented evaluation methodologies is neither comparable to our evaluation method nor between them. This corroborates our initial claim that there is a lack of comparative work in the field which would help an ontology engineer to choose the appropriate methods. One step in this direction is the work in [1]. They present an interesting framework and a corresponding workbench - Mo'K - allowing users to design conceptual clustering methods to assist them in an ontology building task. In particular, they compare different representations, pruning parameters and distance measures. Though their framework is in principle general enough to integrate different clustering methods, they only present results for an agglomerative clustering algorithm.

6 Conclusion and Outlook

We have presented a comparison of different clustering approaches with regard to the task of automatically learning taxonomies from textual data. Such a comparison is novel and to our knowledge has not been presented before. Our conclusion is that the approaches we examine have a comparable performance regarding the task we consider. Regardless of performance, each approach has its own benefits. The novel approach based on FCA we present has the advantage that it not only produces clusters, but also intensional descriptions of these clusters thus facilitating their understanding. The benefit of using Bi-Section-K-Means is certainly its efficiency. Furthermore, we have also proposed a systematic way of evaluating ontology learning algorithms by comparing them to a given human-modeled ontology. In this sense our aim has also been to establish a baseline for further research.

Acknowledgements. Research reported in this paper has been partially financed by EU in the IST projects Dot-Kom (IST-2001-34038) and SWAP (IST-2001-34103).

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*, (2000).

[2] S.A. Caraballo, 'Automatic construction of a hypernym-labeled noun hierarchy from text', in *Proceedings of the 37th Annual Meeting of the ACL*, (1999).

[3] P. Cimiano, A. Hotho, and S. Staab, 'Clustering ontologies from text', in *Proceedings of LREC'04*, (2004).

[4] W. Day and H. Edelsbrunner, 'Efficient algorithms for agglomerative hierarchical clustering methods', *J. of Classification*, **1**(7), (1984).

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

[6] 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. Verlardi, (1998).

[7] D. Fisher, 'Knowledge acquisition via incremental conceptual clustering', *Machine Learning*, (2), 139–172, (1987).

[8] B. Ganter and R. Wille, *Formal Concept Analysis – Mathematical Foundations*, Springer Verlag, 1999.

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

[10] D. Hindle, 'Noun classification from predicate-argument structures', in *Proceedings of the Annual Meeting of the ACL*, pp. 268–275, (1990).

[11] A. Maedche and S. Staab, 'Measuring similarity between ontologies', in *Proceedings of EKAW'02*. Springer, (2002).

[12] F. Pereira, N. Tishby, and L. Lee, 'Distributional clustering of english words', in *Proceedings of the 31st Annual Meeting of the ACL*, (1993).

[13] M. Steinbach, G. Karypis, and V. Kumar, 'A comparison of document clustering techniques', in *KDD Workshop on Text Mining*, (2000).