Ontology-based Text Clustering

A. Hotho and S. Staab

Institute AIFB
University of Karlsruhe
D-76128 Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de/WBS/

A. Maedche

FZI Research Center for
Information Technologies
Haid-und-Neu-Strasse 10-14
D–76131 Karlsruhe, Germany
http://www.fzi.de/wim

Abstract

Text clustering typically involves clustering in a high dimensional space, which appears difficult with regard to virtually all practical settings. In addition, given a particular clustering result it is typically very hard to come up with a good explanation of why the text clusters have been constructed the way they are. In this paper, we propose a new approach for applying background knowledge during preprocessing in order to improve clustering results and allow for selection between results. We built various views basing our selection of text features on a heterarchy of concepts. Based on these aggregations, we compute multiple clustering results using K-Means. The results may be distinguished and explained by the corresponding selection of concepts in the ontology. Our results compare favourably with a sophisticated baseline preprocessing strategy.

1 Introduction

With the abundance of text documents through World Wide Web and corporate document management systems, the dynamic partitioning of texts into previously unseen categories ranks top on the priority list for all business intelligence systems. However, current text clustering approaches still suffer from major problems that greatly limit their practical applicability.

First, text clustering is mostly seen as an *objective* method, which delivers one clearly defined result, which needs to be "optimal" in some way. This, however, runs contrary to the fact that different people have quite different needs with regard to clustering of texts, because they may view the same documents from completely different perspectives (e.g., a business view vs. a technical view; also cf. [9]). Thus, what is needed are *subjective* criteria that allow for a diversity of views from which to look at the clustering task.

Second, text clustering typically is a clustering task working in a *high-dimensional space* where each word is seen as a potential attribute for a text. Empirical and mathematical analysis, however, has shown that — in addition to computational ineffiencies — clustering in high-dimensional spaces is very difficult, because every data point tends to have the same distance from all other data points [2].

Third, text clustering *per se* is often rather useless, unless it is combined with an *explanation* of why particular texts were categorized into a particular cluster. I.e. one output desired from clustering in practical settings is the explanation of why a particular cluster result was produced rather than the result itself. A common method for producing explanations is the learning of rules based on the cluster results. Again, however, this approach suffers from the high number of features chosen for computing clusters.

Though there are of course different approaches for clustering, simple ones like K-Means or sophisticated ones (like [3]), based on the consideration just mentioned we found that virtually all algorithms working on large feature vectors will eventually face the same principal problems without really approaching the matters of *subjectivity* and *explainability*. Hence, our aim was to consider different views onto the data, i.e. different aggregation levels at which text documents may be represented and from which clustering results are eventually derived.

The principal idea of our approach, *COSA* (Concept Selection and Aggregation), is to use a simple, core ontology for restricting the set of relevant document features and for automatically proposing good aggregations. The aggregations are then exploited by the standard clustering algorithm K-Means. More precisely, we have compiled a heterarchy of concepts[1] that is used by a heuristic search

[1] A heterarchy of concepts is a kind of "taxonomy"

algorithm to automatically construct a set of views. The basic criteria of COSA include the computation of support for particular concepts and the top-down navigation of the heterarchy in a greedy manner. Based on COSA, a set of clustering results is produced without interaction by a human user of the system. The user may then decide to prefer the one over the other clustering result based on the actual concepts used for clustering as well as on standard quality measures (such as the silhouette measure [8]).

In this paper, we briefly formalize our notion of ontology (Section 2). We continue describing COSA as well as two baseline preprocessing strategies (Section 3). These three are compared in our experimental evaluation (Section 4), before we relate to other work and conclude.

2 Heterarchy and Core Ontology

A core ontology in our framework is defined by:

Definition 1 (Core Ontology) *A core ontology is a sign system* $\mathcal{O} := (\mathcal{L}, \mathcal{F}, \mathcal{C}, \mathcal{H}, \text{ROOT})$*, which consists of*

- *A* ***lexicon****: The lexicon* $\mathcal{L}$ *contains a set of terms.*
- *A set of* ***concepts*** $\mathcal{C}$*.*
- *The* ***reference function*** $\mathcal{F}$ *with* $\mathcal{F} : 2^{\mathcal{L}} \mapsto 2^{\mathcal{C}}$*.* $\mathcal{F}$ *links sets of terms* $\{L_i\} \subset \mathcal{L}$ *to the set of concepts they refer to. In general, one term may refer to several concepts and one concept may be refered to by several terms (e.g., "boat hire" and "boat rental" may refer to the concept* BOATHIRE*) . The inverse of* $\mathcal{F}$ *is* $\mathcal{F}^{-1}$*.*
- *A* ***heterarchy*** $\mathcal{H}$*: Concepts are taxonomically related by the directed, acyclic, transitive, reflexive relation* $\mathcal{H}$*,* $(\mathcal{H} \subset \mathcal{C} \times \mathcal{C})$*.* $\mathcal{H}(\text{HOTEL}, \text{ACCOMODATION})$ *means that* HOTEL *is a subconcept of* ACCOMODATION*.*
- *A top concept* $\text{ROOT} \in \mathcal{C}$*. For all* $C \in \mathcal{C}$ *it holds:* $\mathcal{H}(C, \text{ROOT})$*.*

The core ontology defines the background knowledge used for preprocessing and selection of relevant views (i.e. aggregations) onto the set of texts. The formulation we have used here roughly corresponds to the basic structures used in the famous WordNet [11], but the actual ontology we have used is domain-specific rather than general as WordNet. In order to easily build new ontologies we have developed a rich framework, which includes support for semi-automatic constructing of ontologies from various input (cf. [10]).

where each term may have multiple parents and — of course — multiple children.

3 Document Preprocessing

Documents may be represented by a wide range of different feature descriptions. The most straightforward description of documents relies on term vectors. A term vector for one document specifies how often each term from the document set occurs in that document. The immediate drawback of this basic approach for document preprocessing for clustering is the size of the feature vectors. In our example evaluation, the feature vectors computed by this method were of size 46,947, which made clustering inefficient and difficult in principle, as described above.

While for supervised learning tasks there exist quite a number of evaluations of how document preprocessing strategies perform (cf., e.g., [4]), there are only few corresponding results for unsupervised knowledge discovery tasks like document clustering (cf. Section 5).

To evaluate our approach, which takes advantage of the background knowledge we provide with our core ontology, we did of course compare against that basic approach for document preprocessing (referred to by *Simple Vector Representation* or *SiVeR* in the following). We were aware that due to the problems with clustering in high dimensional space, SiVeR would be handicapped from the very beginning. In order to perform a more competitive comparison, we have decided to include another preprocessing approach in the evaluation.

Hence, in the following we develop, *(i)*, a preprocessing strategy (cf. Section 3.1) based on term vectors reduced to terms considered "important" by information retrieval measures, *viz.* a preprocessing strategy based on term selection; *(ii)*, a more comprehensive approach using the background knowledge available in the ontology. In particular, we apply techniques from natural language processing to map terms to concepts (cf. Section 3.2) and select between aggregations navigating top-down in the heterarchy.

3.1 Preprocessing Strategy: Term Selection (TES)

Term selection, the second approach we use here for preprocessing, is based on the feature vectors from SiVeR, but focuses on few terms, hence, it produces a low dimensional representation. Selection of terms is based on the information retrieval measure tf/idf:

Definition 2 (tf/idf) *Let* $tf(i, j)$ *be the term frequency of term* j *in a document* $d_i \in \mathcal{D}, i = 1, \ldots, N$*. Let df(j) be the document frequency of term* j *that counts in how many documents term* j *appears. Then tf/idf (term frequency / inverted document frequency of term* j *in document* i *is defined*

by:

$$tfidf(i,j) = tf(i,j) * \log\left(\frac{N}{df(j)}\right). \quad (1)$$

Tf/idf weighs the frequency of a term in a document with a factor that discounts its importance when it appears in almost all documents. Therefore terms that appear too rarely or too frequently are ranked lower than terms that hold the balance and, hence, are expected to be better able to contribute to clustering results.

For TES, we produce the list of all terms contained in one of the documents from the corpus $\mathcal{D}$ except of terms that appear in a standard list of stopwords. Then, TES selects the d best terms j that maximize $W(j)$,

$$W(j) := \sum_{i:=1\ldots N} tfidf(i,j), \quad (2)$$

and produces a d dimensional vector for document d_i containing the tf/idf values, tfidf(i,j), for the d best terms.

3.2 Preprocessing Strategy: Concept Selection and Aggregation (COSA)

Our approach for preprocessing, concept selection and aggregation (COSA), involves two stages. First, COSA maps terms onto concepts using a shallow and efficient natural language processing system. Second, COSA uses the concept heterarchy to propose good aggregations for subsequent clustering.

Mapping terms to concepts

The mapping of terms to concepts relies on some modules of SMES (Saarbrücken Message Extraction System), a shallow text processor for German (cf. [12]). SMES components exploited by COSA comprise a *tokenizer* based on regular expressions and a *lexical analysis* component including a *word* and a *domain lexicon*.

The tokenizer scans the text in order to identify boundaries of words and complex expressions like "$20.00" or "United States of America", and to expand abbreviations. The word lexicon contains more than 120,000 stem entries. Lexical analysis uses the word lexicon, *(i)*, to perform morphological analysis of terms, i. e. the identification of the canonical common stem of a set of related word forms and the analysis of compounds and, *(ii)*, to recognize named entities. Thus, $\mathcal{L}$ as described in Definition 1 is a set defined by the tokenizer, the word lexicon and the analysis procedures of the lexical analysis component. The domain lexicon contains the mappings from word stems to concepts, i.e. together with the other modules it represents the function $\mathcal{F}$ as defined in Definition 1. By this way, e.g., the expression "Hotel Schwarzer Adler" is associated with the concept HOTEL.

Based on this input, each document is represented by a vector of concepts, each entry specifying the frequency that a concept occurs in the document.

A heuristic for generating good aggregations

Because synonyms are mapped to common concepts and because in all realistic document sets there are more terms than concepts, the sizes of concept vectors representing documents are already considerably smaller than the sizes of term vectors produced by SiVeR. Still, realistic settings require at least some hundreds or thousands of concepts, which yields simply too many dimensions for practical clustering.

Therefore, we have looked for ways to heuristically reduce the number of features and, thus, the number of dimensions of the clustering space. The principal idea of our algorithm `GenerateConceptViews` is to navigate the heterarchy top-down and build views from the concept parts that achieve good overall support (cf. Algorithm 1 below).

The variable $Agenda$ is defined to describe the current features used in concept vectors, hence the current view onto the document set. For instance, the current agenda could be [ACCOMODATION, VACATION, SIGHT-SEEING]. A view is altered, by taking the frontmost, i.e. the concept with the most support, from the agenda (lines 4 and 5) and branching — if it is not a leaf concept — into its subconcepts (line 10). In order to restrict branching, we only perform binary splits at a time. Continuing the example just given, the first feature for the input space is described by the concept ACCOMODATION and when ACCOMODATION has the subconcepts [HOTEL, GUEST-HOUSE, YOUTH-HOSTEL], we will select the subconcept that has the highest support of these three (line 11), e.g. HOTEL, and aggregate the other two subconcepts into one feature, viz. [GUEST-HOUSE, YOUTH-HOSTEL] (line 12). The list [GUEST-HOUSE, YOUTH-HOSTEL] is then treated almost like a proper, atomic concept. HOTEL and [GUEST-HOUSE, YOUTH-HOSTEL] are both inserted into $Agenda$ ordering all elements according to their support (lines 13-14). The result might be, e.g., [VACATION, [GUEST-HOUSE, YOUTH-HOSTEL], HOTEL, SIGHT-SEEING].

Thereby, *direct support* of a concept C in a document d_i is defined by the concept frequency cf(i,C) that one of the terms $\mathcal{F}^{-1}(\{C\})$ appears in d_i. Complete support includes also consideration of all the subconcepts:

$$\text{Support}(i,C) := \sum_{B\in\{B|\mathcal{H}(B,C)\}} \text{cf}(i,B), \quad (3)$$

and

$$\text{Support}(C) := \sum_{i:=1\ldots N} \text{Support}(i, C). \quad (4)$$

If the $agenda$ has length $d+1$ due to the last binary split of one of its elements, $agenda$ is shortened by the element with least support (line 15). If the $agenda$ has the correct number of features, it is added to the output set describing a selection of concepts, hence an aggregation that represents documents by d-dimensional concept vectors (line 17).

Thus, Algorithm 1 zooms into those concepts that exhibit strongest support, while taking into account the support of subconcepts. Finally, it proposes sets of views for clustering that imply a d-dimensional representation of documents by concept vectors. Each entry of a vector specifies how often the concept (or its subconcepts) appears in the corresponding document.

3.3 A note on absolute vs. logarithmic values

The document representations described so far used absolute frequency values for concepts or terms (possibly weighted by idf). Considering that the occurrence of terms forms a hyperbolic distribution and, hence, most terms appear only rarely, using the logarithmic value $\log(x+1)$ instead of the absolute value x itself seemed reasonable to improve clustering results. Indeed, for all preprocessing strategies given here, we found that results were only improved compared to absolute values. Hence, all results presented subsequently assume the logarithmic representation of term or concept frequencies.

4 Evaluation

This section describes the evaluation of applying K-Means to the preprocessing strategies SiVeR, TES, and COSA introduced above.

Setting

We have performed all evaluations on a document set from the tourism domain. For this purpose, we have manually modeled an ontology $\mathcal{O}$ consisting of a set of concepts $\mathcal{C}$ ($\mid \mathcal{C} \mid = 1030$), and a word lexicon consisting of 1950 stem entries (the coverage of different terms $\mathcal{L}$ by SMES is much larger!). The heterarchy $\mathcal{H}$ has an average depth of 4.6, the longest uni-directed path from root to leaf is of length 9.

Our document corpus $\mathcal{D}$ has been crawled from a WWW provider for tourist information (URL: http://www.all-in-all.de) consisting now of 2234 HTML documents with a total sum of over 16 million words. The documents in this corpus describe actual objects, like locations, accomodations, facilities of accomodations, administrative information, and cultural events.

Silhouette Coefficient

Our aim was to compare SiVeR, TES, and COSA for a wide range of parameter settings. In order to be rather independent from the number of features used for clustering and the number of clusters produced as result, our main comparisons refer to the silhouette coefficient (cf.[8]):

Definition 3 (Silhouette Coefficient) *Let $D_M = \{\overline{D}_1, \ldots, \overline{D}_k\}$ describe a clustering result, i.e. it is an exhaustive partitioning of the set of documents $\mathcal{D}$. The distance of a document $d \in \mathcal{D}$ to a cluster $\overline{D}_i \in D_M$ is given as*

$$dist(d, \overline{D}_i) = \frac{\sum_{p \in \overline{D}_i} dist(d, p)}{\mid \overline{D}_i \mid}. \quad (5)$$

Let further be $a(d, D_M) = dist(d, \overline{D}_l)$ the distance of document d to its cluster $\overline{D}_l$ ($d \in \overline{D}_l$), and $b(d, D_M) = \min_{\overline{D}_i \in D_M, d \notin \overline{D}_i} dist(d, \overline{D}_i)$ the distance of document d to the nearest neighbouring cluster.

*The **silhouette** $s(d, D_M)$ of a document $d \in \mathcal{D}$ is then defined as:*

$$s(d, D_M) = \frac{b(d, D_M) - a(d, D_M)}{\max\{a(d, D_M), b(d, D_M)\}}. \quad (6)$$

*The **silhouette coefficient** $SC(D_M)$ as:*

$$SC(D_M) = \frac{\sum_{p \in \mathcal{D}} s(p, D_M)}{\mid \mathcal{D} \mid}. \quad (7)$$

The silhouette coefficient is a measure for the clustering quality, that is rather independent from the number of clusters, k. Experiences, such as documented in [8], show that values between 0.7 and 1.0 indicate clustering results with excellent separation between clusters, *viz.* data points are very close to the center of their cluster and remote from the next nearest cluster. For the range from 0.5 to 0.7 one finds that data points are clearly assigned to cluster centers. Values from 0.25 to 0.5 indicate that cluster centers can be found, though there is considerable "noise", i.e. there are many data points that cannot be clearly assigned to clusters. Below a value of 0.25 it becomes practically impossible to find significant cluster centers and to definitely assign the majority of data points.

For comparison of the three different preprocessing methods we have used standard K-Means. However, we are well aware that for high-dimensional data approaches like [3] may improve results — very likely for all three preprocessing strategies. However, in preliminary tests we found that in the low-dimensional realms where the silhouette coefficient indicated reasonable separation

Algorithm 1 (`GenerateConceptViews`)
Input: *number of dimensions d, Ontology $\mathcal{O}$ with top concept* ROOT, *document set $\mathcal{D}$*

```
1  begin
2    Agenda := [ROOT];
3    repeat
4      Elem := First(Agenda);
5      Agenda := Rest(Agenda);
7      if Leaf(Elem)
8        then continue := FALSE;
9        else
10           if Atom(Elem) then Elem := Subconcepts(Elem); fi;
11           NewElem := BestSupportElem(Elem);
12           RestElem := Elem \ NewElem;
13           if ¬Empty(RestElem) then Agenda := SortInto(RestElem, Agenda); fi;
14           Agenda := SortInto(NewElem, Agenda);
15           if Length(Agenda) > d then Agenda := Butlast(Agenda); fi;
16     fi;
17     if Length(Agenda) = d then Output(Agenda); fi;
18   until continue = FALSE;
19 end
```

Output: *Set of lists consisting of single concepts and lists of concepts, which describe views onto the document corpus $\mathcal{D}$.*
Auxiliary functions used:

Subconcepts(C)	*returns an arbitrarily ordered list of direct subconcepts of C.*
Support(C)	*cf. equation 4.*
Support$(ListC)$	*is the sum over all concepts C in $ListC$ of Support(C).*
SortInto$(Element, List2)$	*sorts $Element$, which may be a single concept or a list of concepts, as a whole into $List2$ ordering according to Support$(Element)$ and removing redundant elements.*
BestSupportElem(List)	*returns the $Element$ of $List$ with maximal Support$(Element)$.*
$[Element]$	*constructs list with one $Element$.*
$[Element, List]$	*list constructor extending $List$ such that $Element$ is first.*
First$(List)$, *Rest*$(List)$	*are the common list processing functions.*
Atom(E)	*returns true when E is not a list.*
Leaf(E)	*returns true when E is a concept without subconcepts.*
$List \setminus E$	*removes element E from $List$.*
Length$(List)$	*returns the length of $List$.*
Butlast$(List)$	*returns a list identical to $List$, but excluding the last element.*

between clusters, quality measures for standard and improved K-Means coincided.

The general result of our evaluation using the silhouette measure was that K-Means based on COSA preprocessing excelled the comparison baseline, viz. K-Means based on TES, to a large extent. K-Means based on SiVeR was so strongly handicapped by having to cope with overly many dimensions that its silhouette coefficient always approached 0 — indicating that no reasonable clustering structures could be found.

One exemplary, but overall characteristic diagramm depicted in Figure 1 shows the silhouette coefficient for a fixed number of features used (namely 15) and a fixed number of clusters produced (namely 10). It does so for K-Means based on SiVeR, for K-Means based on TES, and for K-Means based on COSA. The results for SiVeR are strictly disappointing. TES is considerably better, but it still yields a silhouette coefficient that indicates practically non-existent distinctions between clusters. COSA produces for this parameter setting 89 views. We found that the best aggregations produced from COSA delivered clustering results with silhouette measures of up to 0.48 — indicating indeed very reasonable separation between clusters.

Mean Squared Error

To base our comparisons not only on one single measure we also did a study, evaluating the mean squared error (MSE) for our approach and the corresponding baseline results. The mean squared error is a measure of compactness of a given clustering and is defined as follows.

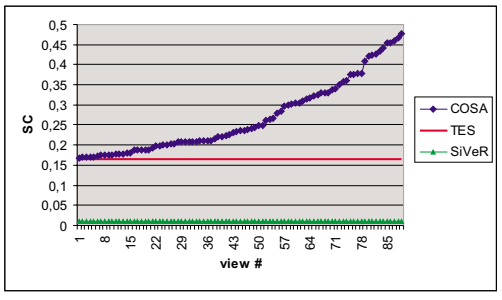

Figure 1: Comparing TES with 89 views produced by COSA for $k = 10;\ d = 15$.

Definition 4 (MSE) *The overall mean squared error MSE for a given clustering $D_M = \{\overline{D}_1, \ldots, \overline{D}_k\}$ is defined as*

$$MSE(D_M) = \sum_{i=1}^{k} MSE(\overline{D}_i). \qquad (8)$$

The mean squared error for one cluster $MSE(\overline{D}_i)$ is defined as

$$MSE(\overline{D}_i) = \sum_{p \in \overline{D}_i} dist(p, \mu_{\overline{D}_i})^2. \qquad (9)$$

(where $\mu_{\overline{D}_i}$ is the centroid of cluster $\overline{D}_i$)

We found that also for the MSE measure K-Means based on COSA compared favorably against K-Means based on TES. 49 of COSA's views are worse, but 40 of them are considerably better than the TES baseline. Figure 2 shows the corresponding results with a baseline for TES at 3240, but the best values for COSA exhibit a MSE of only 1314.

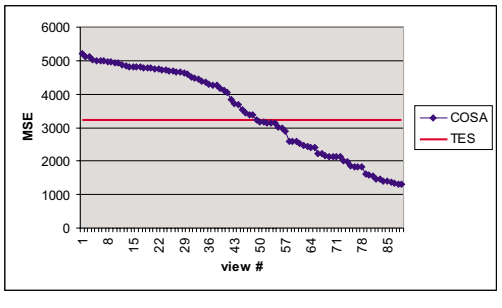

Figure 2: Comparing TES and 89 views produced by COSA; $k = 10;\ d = 15$.

Varying number of features d and clusters k

Then we explored how COSA and TES would fare when varying the number of features used and the number of clusters produced by K-Means.

Figure 3 depicts the dependency between the number of features, d, used and the preprocessing method for a fixed number of clusters produced by K-Means, viz. $k = 10$. The line for COSA shows the silhouette coefficient for the best aggregation from the ones generated by `GenerateConceptViews`. We see that for TES and COSA the quality of results decreases — as expected — for the higher dimensions, though COSA still compares favorably against TES.

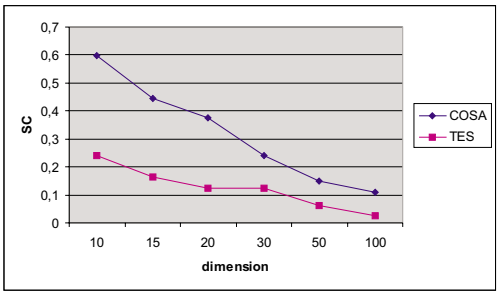

Figure 3: Comparing TES and the best aggregation of COSA; $k = 10;\ d = 10, 15, 30, 50, 100$.

We have not included the lower bound of COSA in Figure 3. The reason is that — so far — we have not been very attentive to optimize `GenerateConceptViews` in order to eliminate the worst views up front. This, however, should be easily possible, because we observed that the bad results are produced by views that contain too many overly general concepts like MATERIALTHING or INTANGIBLE.

In the next experiments, we have varied the number of clusters, k, between 2 and 100, while d remained at 15 (cf. Figure 4). The general result is that the number of clusters does not affect the results produced by COSA and TES very much. The slight increase of the silhouette coefficient is due to a growing number of documents (up to 30%) that cluster exactly at one point. There, viz. at $(0, \ldots, 0)$, all features disappear.

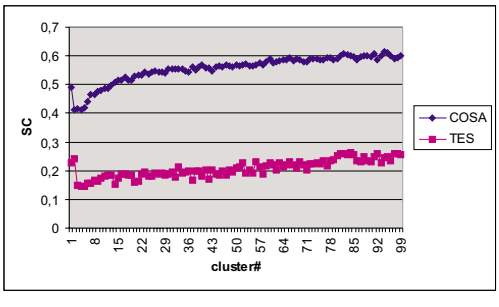

Figure 4: Comparing TES and the best aggregation produced by COSA; $k = 2 \ldots 100;\ d = 15$.

Example for Interpretation

In order to provide a more concrete intuition of the type of results returned by GENERATECONCEPTVIEW, we here show the list of concepts that corresponds to the best aggregation for parameters $k = 10$ and $d = 10$ and in a silhouette coefficient of 0.598:

SAUNA, SOLARIUM, TERRACE, BEACH, SEA_RESSORT, ACTION_AT_OBJECT, OVERNIGHT_STAY, WATER_SPORTS, TRAVELING, HOTEL_CATEGORY

Comparing some plain lists may already give the user an intuition of how clustering results might be distinguishable (or not distinguishable if the views are very similar!). A better grip at interpretation is however achieved by depicting the relevant parts of the heterarchy as shown in Figure 5.

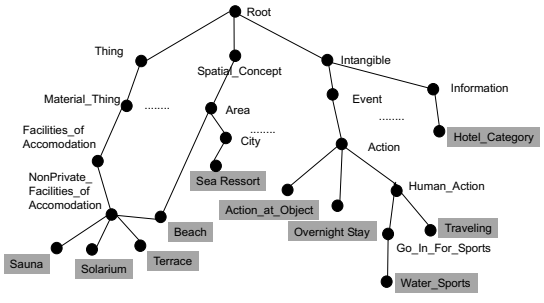

Figure 5: An example view generated by COSA

Here, one may recognize that NONPRIVATE_FACILITIES_OF_ACCOMODATION and ACTIONs were important concepts used for clustering and distinguishing documents. Interpreting results, we may conjecture that HOTEL_CATEGORY (three, four, five star hotels, etc.) is a concept, which might perhaps correlate with facilities of the accomodation — a correlation that happens to be not described in the given ontology. Finally, we see SEA RESSORT in this view, which might play a role for clustering or which might occur just because of uninterpretable "noise".

We currently explore GUI possibilities in order to tie the interpretation of clustering results with the navigation of the heterarchy in order to give the user a good grip at different clustering views.

Conclusion

Our results support the general statement that structure can mostly be found in a low dimensional space (cf. [2]). Our proposal is well suited to provide a selected number of aggregations in subspaces exploiting standard K-Means and comparing favorably with baselines, like clustering based on d terms ranked by tf/idf measures.

The selected concepts may be used to indicate to the user, which text features were most relevant for the particular clustering results and to distinguish different views.

5 Related Work

All clustering approaches based on frequencies of terms/concepts and similarities of data points suffer from the same mathematical properties of the underlying spaces (cf. [2; 5]). These properties imply that even when "good" clusters with relatively small mean squared errors can be built, these clusters do not exhibit significant structural information as their data points are not really more similar to each other than to many other data points. Therefore, we derive the high-level requirement for text clustering approaches that they either rely on much more background knowledge (and thus can come up with new measures for similarity) or that they cluster in subspaces of the input space.

In general, existing approaches (e.g., [1; 6]) on subspace clustering face the dual nature of "good quality". On the one hand, there are sound statistical measures for judging quality. State-of-the-art methods use them in order to produce "good" projections and, hence, "good" clustering results, for instance:

- Hinneburg & Keim [6] show how projections improve the effectiveness and efficiency of the clustering process. Their work shows that projections are important for improving the performance of clustering algorithms. In contrast to our work, they do not focus on cluster quality with respect to the internal structures contained in the clustering.
- The problem of clustering high-dimensional data sets has been researched by Agrawal et al. [1]: They present a clustering algorithm called CLIQUE that identifies dense clusters in subspaces of maximum dimensionality. Cluster descriptions are generated in the form of minimized DNF expressions.
- A straightforward preprocessing strategy may be derived from multivariate statistical data analysis known under the name principal component analysis (PCA). PCA reduces the number of features by replacing a set of features by a new feature representing their combination.
- In [14], Schuetze and Silverstein have researched and evaluated projection techniques for efficient document clustering. They show how different projection techniques significantly improve performance for clustering, not accompanied by a loss of cluster quality. They distinguish between local and global projection, where local projection maps each document onto a different subspace, and, global projection selects the relevant terms for all documents using latent semantic indexing.

Now, on the other hand, in real-world applications the statistically optimal projection, such as used in the approaches just cited, often does not coincide with the projection most suitable for humans to solve a particular task, such as finding the

right piece of knowledge in a large set of documents. Users typically prefer explicit background knowledge that indicates the foundations on which a clustering results has been achieved.

Hinneburg et al. [7] consider this general problem a domain specific optimization task. Therefore, they propose to use a visual and interactive environment to derive meaningful projections involving the user. Our approach may be seen to solve some part of task they assign to the user environment automatically, while giving the user some first means to explore the result space interactively in order to select the projection most relevant for her particular objectives.

Finally, we want to mention an interesting proposal for feature selection made in [13]. Devaney and Ram describe feature selection for an unsupervised learning task, namely conceptual clustering. They discuss a sequential feature selection strategy based on an existing COBWEB conceptual clustering system. In their evaluation they show that feature selection significantly improves the results of COBWEB. The drawback that Devaney and Ram face, however, is that COBWEB is not scalable like K-Means. Hence, for practical purposes of clustering in large document repositories, COSA seems better suited.

6 Conclusion

In this paper we have shown how to include background knowledge in form of a heterarchy in order to generate different clustering views onto a set of documents. We have compared our approach against a sophisticated baseline, achieving a result favourable for our approach. In addition, we have shown that it is possible to automatically produce results for diverging views onto the same input. Thereby, the user can rely on a heterarchy to control and possibly interpret clustering results.

The preprocessing method, COSA, that we propose is a very general one. Currently we try to apply our techniques on a highly-dimensional data set that is not based on text documents, but on a real-world customer data base with over 3 million clients. First preliminary results of our method on this data base are very encouraging.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data, Seattle, Washington, June 1998*. ACM Press, 1998.

[2] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is 'nearest neighbor' meaningful. In *Proc. of ICDT-1999, Jerusalem, Israel, 1999*, pages 217–235, 1999.

[3] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. of KDD-1998, New York, NY, USA, August 1998*, pages 9–15, Menlo Park, CA, USA, 1998. AAAI Press.

[4] J. Fuernkranz, T. Mitchell, and E. Riloff. A Case Study in Using Linguistic Phrases for Text Categorization on the WWW. In *Proc. of AAAI/ICML Workshop Learning for Text Categorization, Madison, WI, 1998*. AAAI Press, 1998.

[5] A. Hinneburg, C. Aggarwal, and D.A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proc. of VLDB-2000, Cairo, Egypt, September 2000*, pages 506–515. Morgan Kaufmann, 2000.

[6] A. Hinneburg and D.A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. of VLDB-1999, Edinburgh, Scotland, September 2000*. Morgan Kaufmann, 1999.

[7] A. Hinneburg, M. Wawryniuk, and D.A. Keim. Visual mining of high-dimensional data. *Computer Graphics & Applications Journal*, September 1999.

[8] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[9] S. A. Macskassy, A. Banerjee, B.D. Davison, and H. Hirsh. Human performance on clustering web pages: a preliminary study. In *Proc. of KDD-1998, New York, NY, USA, August 1998*, pages 264–268, Menlo Park, CA, USA, 1998. AAAI Press.

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

[11] G. Miller. WordNet: A lexical database for english. *CACM*, 38(11):39–41, 1995.

[12] G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An information extraction core system for real world german text processing. In *ANLP-1997 — Proceedings of the Conference on Applied Natural Language Processing*, pages 208–215, Washington, USA, 1997.

[13] M. Devaneyand A. Ram. Efficient feature selection in conceptual clustering. In *Proc. of ICML-1997, Nashville, TN, 1998*. Morgan Kaufmann, 1998.

[14] H. Schuetze and C. Silverstein. Projections for efficient document clustering. In *Proc. of SIGIR-1997, Philadelphia, PA, July 1997*, pages 74–81. Morgan Kaufmann, 1997.