Frequent Term-Based Text Clustering

Florian Beil
Institute for Computer Science Ludwig-Maximilians-Universitaet Muenchen
80538 Munich, Germany
beil@informatik.uni-muenchen.de

Martin Ester
School of Computing Science
Simon Fraser University
Burnaby BC
Canada V5A 1S6
ester@cs.cfu.ca

Xiaowei Xu
Information and Comunications
Corporate Technology
Siemens AG
81730 Munich, Germany
Xiaowei.Xu@mchp.siemens.de

ABSTRACT
Text clustering methods can be used to structure large sets of text or hypertext documents. The well-known methods of text clustering, however, do not really address the special problems of text clustering: very high dimensionality of the data, very large size of the databases and understandability of the cluster description. In this paper, we introduce a novel approach which uses frequent item (term) sets for text clustering. Such frequent sets can be efficiently discovered using algorithms for association rule mining. To cluster based on frequent term sets, we measure the mutual overlap of frequent sets with respect to the sets of supporting documents. We present two algorithms for frequent term-based text clustering, FTC which creates flat clusterings and HFTC for hierarchical clustering. An experimental evaluation on classical text documents as well as on web documents demonstrates that the proposed algorithms obtain clusterings of comparable quality significantly more efficiently than state-of-the-art text clustering algorithms. Furthermore, our methods provide an understandable description of the discovered clusters by their frequent term sets.

Keywords
Clustering, Frequent Item Sets, Text Documents.

1. INTRODUCTION
The world wide web continues to grow at an amazing speed. On the other hand, there is also a quickly growing number of text and hypertext documents managed in organizational intranets, representing the accumulated knowledge of organizations that becomes more and more important for their success in today's information society. Due to the huge size, high dynamics, and large diversity of the web and of organizational intranets, it has become a very challenging task to find the truly relevant content for some user or purpose. For example, the standard web search engines have low precision, since typically a large number of irrelevant web pages is returned together with a small number of relevant pages. This phenomenon is mainly due to the fact that keywords specified by the user may occur in different contexts, consider for example the term "cluster". Consequently, a web search engine typically returns long lists of results, but the user, in his limited amount of time, processes only the first few results. Thus, a lot of truely relevant information hidden in the long result lists will never be discovered. Text clustering methods can be applied to structure the large result set such that they can be interactively browsed by the user. Effective knowledge management is a major competitive advantage in today's information society. To structure large sets of hypertexts available in a company's intranet, again methods of text clustering can be used.

Compared to previous applications of clustering, three major challenges must be addressed for clustering (hyper)text databases (see also [1]):

- Very high dimensionality of the data (~ 10,000 terms / dimensions): this requires the ability to deal with sparse data spaces or a method of dimensionality reduction.
- Very large size of the databases (in particular, of the world wide web): therefore, the clustering algorithms must be very efficient and scalable to large databases.
- Understandable description of the clusters: the cluster descriptions guide the user in the process of browsing the clustering and, therefore, they must be understandable also to non-experts.

A lot of different text clustering algorithms have been proposed in the literature, including Scatter/Gather [2], SuffixTree Clustering [3] and bisecting k-means [4]. A recent comparison [4] demonstrates that bisecting k-means outperforms the other well-known techniques, in particular hierarchical clustering algorithms, with respect to clustering quality. Furthermore, this algorithm is efficient. However, bisecting k-means like most of the other algorithms does not really address the above mentioned problems of text clustering: it clusters the full high-dimensional vector space of term frequency vectors and the discovered means of the clusters do not provide an understandable description of the documents grouped in some cluster.

In this paper, we present a novel approach which uses frequent item (term) sets for text clustering. Frequent term sets are sets of terms co-occurring in more than a threshold percentage of all documents of a database. Such frequent sets can be efficiently discovered using algorithms such as Apriori [5]. This approach allows us to reduce drastically the dimensionality of the data, is efficient even for very large databases, and provides an understandable description of the discovered clusters by their frequent term sets.

The rest of this paper is organized as follows. Section 2 briefly introduces the general approach of text clustering including preprocessing and discusses the well-known algorithms. In section 3, we introduce our novel method of frequent term-based text clustering and present the algorithms FTC and HFTC. An experimental evaluation on real text data was conducted, and section 4 reports its major results. Section 5 summarizes the paper and outlines some interesting directions for future research.

2. OVERVIEW ON TEXT CLUSTERING

All methods of text clustering require several steps of preprocessing of the data. First, any non-textual information such as HTML-tags and punctuation is removed from the documents. Then, stopwords such as "I", "am", "and" etc. are also removed. A *term* is any sequence of characters separated from other terms by some delimiter. Note that a term may either be a single word or consist of several words. Typically, the terms are reduced to their basic stem applying a stemming algorithm.

Most text clustering algorithms rely on the so-called *vector-space model*. In this model, each text document d is represented by a vector of frequencies of the remaining m terms:

$$d = (tf_1, \ldots, tf_m).$$

Often, the document vectors are normalized to unit length to allow comparison of documents of different lengths. Note that the vector-space has a very high dimensionality since even after preprocessing there are typically still several thousands of terms, in many text databases you have to deal with approximately 10,000 terms. Due to the high dimensionality, most frequencies are zero for any single document.

To measure the similarity between two documents d_1 and d_2 represented in the vector space model, typically the cosine measure is used which is defined by the cosine of the angle between the two vectors:

$$similarity(d_1, d_2) = \frac{(d_1 \bullet d_2)}{\|d_1\| \cdot \|d_2\|}$$

where $\bullet$ denotes the vector dot product and $\| \|$ denotes the length of a vector.

The standard clustering algorithms can be categorized into *partitioning algorithms* such as k-means or k-medoid and *hierarchical algorithms* such as Single-Link or Average-Link [6]. *Scatter/Gather* [2] is a well-known algorithm which has been proposed for a document browsing system based on clustering. It uses a hierarchical clustering algorithm to determine an initial clustering which is then refined using the k-means clustering algorithm. Many variants of the k-means algorithm have been proposed for the purpose of text clustering, e.g. [7], in particular to determine a good initial clustering. A recent study [4] has compared partitioning and hierarchical methods of text clustering on a broad variety of test datasets. It concludes that k-means clearly outperforms the hierarchical methods with respect to clustering quality. Note that k-means is also much more efficient than hierarchical clustering algorithms. Furthermore, a variant of k-means called *bisecting k-means* is introduced, which yields even better performance. Bisecting k-means uses k-means to partition the dataset into two clusters. Then it keeps partitioning the currently largest cluster into two clusters, again using k-means, until a total number of k clusters has been discovered.

The above methods of text clustering algorithms do not really address the special challenges of text clustering: they cluster the full high-dimensional vector-space and the centroids / means of the discovered clusters do not provide an understandable description of the clusters. This has motivated the development of new special text clustering methods which are not based on the vector space model. *SuffixTree Clustering* [3] is a first method following this approach. Its idea is to form clusters of documents sharing common terms or phrases (multi-word terms). Basic clusters are sets of documents containing a single given term. A cluster graph is built with nodes representing basic clusters and edges representing an overlap of at least 50% between the two associated basic clusters. A cluster is defined as a connected component in this cluster graph. The drawback of SuffixTree Clustering is that, while two directly neighboring basic clusters in the graph must be similar, two distant nodes (basic clusters) within a connected component do not have to be similar at all. Unfortunately, Suffix Tree Clustering has not been evaluated on standard test data sets so that its performance can hardly be compared with other methods.

3. FREQUENT TERM-BASED TEXT CLUSTERING

Frequent item sets form the basis of association rule mining. Exploiting the monotonicity property of frequent item sets (each subset of a frequent item set is also frequent) and using data structures supporting the support counting, the set of all frequent item sets can be efficiently determined even for large databases. Many different algorithms have been developed for that task, including Apriori [5]. See [8] for an overview on association rule mining. Frequent item sets can also be used for the task of classification. [9] introduces a general method of building an effective classifier from the frequent item sets of a database. [10] presents a modification of this approach for the purpose of text classification.

A frequent item-based approach of clustering is promising because it provides a natural way of reducing the large dimensionality of the document vector space. Since we are dealing not with transactions but with documents, we will use the notion of *term* sets instead of *item* sets. A term is any preprocessed word within a document, and a document can be considered as a set of terms occurring in that document at least once. The key idea is not to cluster the high-dimensional vector space, but to consider only the low-dimensional frequent term sets as cluster candidates. A well-selected subset of the set of all frequent term sets can be considered as a clustering. Strictly speaking, a frequent term set is not a cluster (candidate) but only the description of a cluster (candidate). The corresponding cluster itself consists of the set of documents containing all terms of the frequent term set. Unlike in the case of classification, there are no class labels to guide the selection of such a subset from the set of all frequent term sets. Instead, we propose to use the mutual overlap of the frequent term sets with respect to their sets of supporting documents (the clusters) to determine a clustering. The rationale behind this approach is that a small overlap of the clusters will result in a small classification error, when the clustering is later used for classifying new documents. In this section, we introduce the necessary definitions and present two algorithms for frequent term-based text clustering.

3.1 Definitions

Let $D = \{D_1, \ldots, D_n\}$ be a *database of text (hypertext) documents* and T be the set of all *terms* occurring in the documents of D. Each document D_j is represented by the set of terms occurring in D_j, i.e. $D_j \subseteq T$. Let *minsupp* be a real number, $0 \leq \text{minsupp} \leq 1$. For any set of terms $S, S \subseteq T$, let *cov(S)* denote the *cover* of S, the set of all documents containing all terms of S, i,e. the set of all documents supporting S. More precisely, $\text{cov}(S) = \{D_j \in D \mid S \subseteq D_j\}$.

Let $F = \{F_1, \ldots, F_k\}$ be the *set of all frequent term sets* in D with respect to *minsupp*, the set of all term sets contained in at least minsupp of the D documents, i.e.

$$F = \{F_i \subseteq T \mid |\text{cov}(F_i)| \geq \text{minsupp} \cdot |D|\}$$

A frequent term set of cardinality k is called a *frequent k-term set.* The cover of each element F_i of F can be regarded as a *cluster (candidate).*

A *clustering* is any subset of the set of all subsets of the database D such that each document of D is contained in at least one of the sets (clusters). The clusters of a clustering may or may not overlap. We define a *clustering description* as a subset CD of F which covers the whole database, i.e. a clustering description $CD = \{F_i \mid i \in I\}$ has to satisfy the condition $\bigcup_{i \in I} \text{cov}(F_i) = D$.

We want to determine clusterings with a minimum overlap of the clusters. For an efficiently to calculate measure of the overlap of a given cluster (description) F_i with the union of the other cluster (description)s, we use the number of frequent term sets supported by the documents covered by F_i,. Let f_j denote the number of all frequent term sets supported by document D_j, i.e.

$$f_j = |\{F_i \in R \mid F_i \subseteq D_j\}|$$

where $|\ |$ denotes the cardinality of some set and R denotes a subset of F, the subset of *remaining frequent term sets*, i.e. the difference of F and the set of the already selected frequent term sets. This definition is motivated by the fact that our clustering algorithm will be a greedy algorithm selecting one next cluster (description, i.e. frequent term set) at a time from the set R of remaining frequent term sets. R is step by step reduced by the next selected cluster description.

The overlap of a cluster C_i with the other clusters is the smaller, the smaller the f_j values of its documents are. Ideally, all its documents support no other cluster candidates, i.e. $f_j = 1$ for all documents of C_i, and then C_i has an overlap of 0 with the other cluster candidates. Thus, we define the *standard overlap* of a cluster C_i, denoted by $SO(C_i)$, as the average f_j value (-1) of a cluster, i.e.

$$SO(C_i) = \frac{\sum_{Dj \in Ci} (f_j - 1)}{|C_i|}.$$

The standard overlap is easy to calculate but it suffers from the following effect. Due to the monotonicity property of frequent term sets (each subset of a frequent term set is also frequent), each document supporting one term set of size m also supports at least m-1 other term sets, one of each the sizes m-1, m-2, . . ., 1. Thus, a cluster candidate described by many terms tends to have a much larger standard overlap than a cluster candidate described by only a few terms. Consequently, the standard overlap favors frequent term sets consisting of a very small number of terms.

An alternative definition of the overlap, based on the entropy, minimizes this effect. The entropy measures the distribution of the documents supporting some cluster over all the remaining cluster candidates. While f_j measures the distribution of document D_j over the cluster candidates (frequent term sets), $p_j = \frac{1}{f_j}$ represents the probability that document D_j belongs to one specific cluster candidate. $p_j = 1$ if $f_j = 1$, i.e. if D_j is contained in only 1 cluster candidate. On the other hand, p_j becomes very small for large f_j values. We define the *entropy overlap* of cluster C_i, denoted by $EO(C_i)$, as the distribution of the documents of cluster C_i over all the remaining cluster candidates, i.e.

$$EO(C_i) = \sum_{Dj \in Ci} -\frac{1}{f_j} \cdot \ln\left(\frac{1}{f_j}\right).$$

The entropy overlap becomes 0, if all documents D_j of C_i do not support any other frequent term set ($f_j = 1$), and it increases monotonically with increasing f_j values. As our experimental evaluation will demonstrate, the entropy overlap yields cluster descriptions with significantly more terms than the standard overlap which results in a higher clustering quality.

3.2 Two Clustering Algorithms

We have defined a (flat) clustering as a subset of the set of all subsets of the database D, described by a subset of the set F of all frequent term sets, that covers the whole database. To discover a clustering with a minimum overlap of the clusters, we follow a greedy approach. This approach is motivated by the inherent complexity of the frequent term-based clustering problem: the number of all subsets of F is $O(2^{|F|})$, and, therefore, an exhaustive search is prohibitive. In this section, we present two greedy algorithms for frequent term-based clustering.

- Algorithm *FTC* (Frequent Term-based Clustering): determines a *flat clustering*, i.e. an unstructured set of clusters covering the whole database.
- Algorithm *HFTC* (Hierarchical Frequent Term-based Clustering): determines a *hierarchical clustering*, i.e. a graph-structured clustering with a subset relationship between each cluster and its predecessors in the hierarchy.

Algorithm FTC works in a bottom-up fashion. Starting with an empty set, it continues selecting one more element (one cluster description) from the set of remaining frequent term sets until the entire database is contained in the cover of the set of all chosen frequent term sets (the clustering). In each step, FTC selects the remaining frequent term set with a cover having the minimum overlap with the other cluster candidates. Note that the documents covered by the selected frequent term set are removed from the database *D* and, in the next iteration, the overlap for all remaining cluster candidates is recalculated with respect to the reduced database. Figure 1 presents algorithm FTC in pseudo-code. `DetermineFrequentTermsets` is any efficient algorithm for finding the set of all frequent term sets in database `D` with respect to a minimum support of `minsup`.

```
FTC(database D, float minsup)
SelectedTermSets:= {};
n:= |D|;
RemainingTermSets:=
DetermineFrequentTermsets(D, minsup);
while |cov(SelectedTermSets)| ≠ n do
    for each set in RemainingTermSets do
        Calculate overlap for set;
    BestCandidate:=element of Remaining
    TermSets with minimum overlap;
    SelectedTermSets:= SelectedTermSets
                       ∪ {BestCandidate};
    RemainingTermSets:= RemainingTermSets
                        -{BestCandidate};
    Remove all documents in
    cov(BestCandidate) from D and from the
    coverage of all of the
    RemainingTermSets;
return SelectedTermSets and the cover
    of the elements of SelectedTermSets;
```

Figure 1: Algorithm FTC

Figure 2 illustrates the first step of algorithm FTC on a sample database consisting of 16 documents. In this step, the cluster described by {sun, beach, fun} is selected because of its minimum (entropy) overlap, and the documents D_8, D_{10}, D_{11}, D_{15} are removed from the database as well as from the remaining cluster candidates.

frequent term set	cluster candidate	EO
{sun}	{D_1, D_2, D_4, D_5, D_6, D_8, D_9, D_{10}, D_{11}, D_{13}, D_{15}}	2.98
{fun}	{D_1, D_3, D_4, D_6, D_7, D_8, D_{10}, D_{11}, D_{14}, D_{15}, D_{16}}	3.0
(beach}	{D_2, D_7, D_8, D_9, D_{10}, D_{12}, D_{13}, D_{14}, D_{15}}	2.85
{surf}	{D_1, D_2, D_6, D_7, D_{10}, D_{11}, D_{12}, D_{14}, D_{16}}	2.73
{sun, fun}	{D_1, D_4, D_6, D_8, D_{10}, D_{11}, D_{15}}	1.97
{fun, surf}	{D_1, D_6, D_7, D_{10}, D_{11}, D_{16}}	1.72
{sun, beach}	{D_2, D_8, D_9, D_{10}, D_{11}, D_{15}}	1.72
{sun, surf}	{D_1, D_2, D_6, D_{10}, D_{11}}	1.34
{fun, beach}	{D_7, D_8, D_{10}, D_{14}, D_{15}}	1.47
{beach, surf}	{D_2, D_7, D_{10}, D_{12}, D_{14}}	1.47
{sun, fun, surf}	{D_1, D_6, D_{10}, D_{11}}	0.98
{sun, beach, fun}	**{D_8, D_{10}, D_{11}, D_{15}}**	**0.9**

Figure 2: Illustration of Algorithm FTC

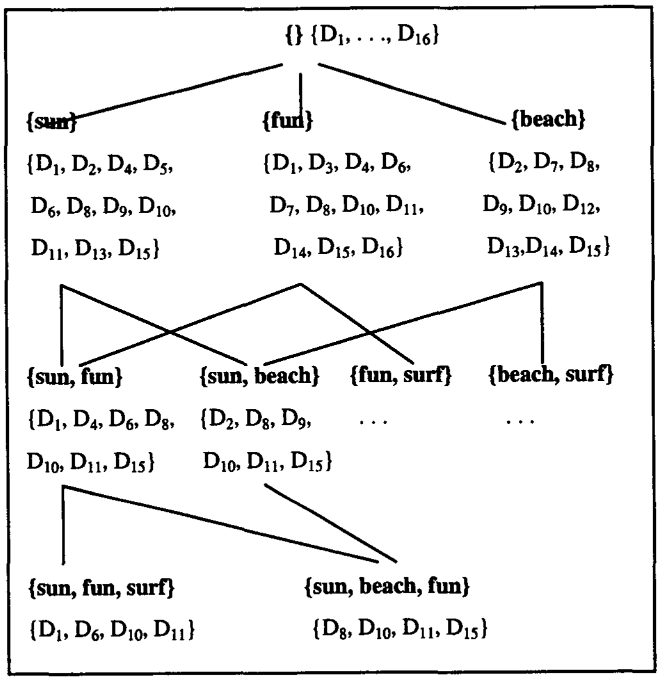
Note that algorithm FTC returns a *clustering description* and a *clustering*. This clustering is non-overlapping. However, algorithm FTC can easily be modified to discover an overlapping clustering by not removing the documents covered by the selected frequent term set from the database *D*.

Due to the monotonicity property, the frequent term sets form a lattice structure: all 1-subsets of frequent 2-sets are also frequent, all 2-subsets of frequent 3-sets are also frequent etc.. This property can be exploited to discover a *hierarchical* frequent term-based clustering. Algorithm *HFTC* (*Hierarchical Frequent Term-based Clustering*) is based on the flat algorithm FTC and applies it in a hierarchical way as follows:

- The algorithm FTC is applied not on the whole set *F* of frequent term sets, but only to the set of all frequent term sets of a single level, i.e. the frequent k-term sets.
- The resulting clusters are further partitioned applying FTC on the frequent term sets of the next level, i.e. using the frequent k+1-term sets. Note that for further partitioning a given cluster, only those frequent k+1-term sets are used which contain the frequent k-term set describing that cluster.

Algorithm HFTC discovers a hierarchical clustering with an empty term set in the root (covering the entire database), using frequent 1-term sets on the first level, frequent 2-term sets on the second level, etc. HFTC stops adding another level to the hierarchical clustering, when there are no frequent term sets for the next level. Furthermore, the successors of some cluster do not have to cover the whole cluster if that is impossible due to a lack of more frequent term sets of this level.

When using the original FTC algorithm, we generate a non-overlapping hierarchical clustering. But when modifying FTC so that it discovers an overlapping flat clustering, we can also discover an overlapping hierarchical clustering. Our experimental evaluation has shown that overlapping hierarchical clusterings are clearly superior with respect to clustering quality, therefore HFTC creates a hierarchical *overlapping* clustering. Figure 3 depicts a part of a hierarchical clustering for our sample database. It consists of three levels since the database does not contain any frequent 4-term sets with respect to the chosen minsup value. The edges denote subset-relationships between the connected clusters.



{} {D1, . . ., D16}
{sun}
{D1, D2, D4, D5, D6, D8, D9, D10, D11, D13, D15}
{fun}
{D1, D3, D4, D6, D7, D8, D10, D11, D14, D15, D16}
{beach}
{D2, D7, D8, D9, D10, D12, D13, D14, D15}
{sun, fun}
{D1, D4, D6, D8, D10, D11, D15}
{sun, beach}
{D2, D8, D9, D10, D11, D15}
{fun, surf}
. . .
{beach, surf}
. . .
{sun, fun, surf}
{D1, D6, D10, D11}
{sun, beach, fun}
{D8, D10, D11, D15}

Figure 3: A hierarchical clustering of the sample database

4. EXPERIMENTAL EVALUATION

FTC and HFTC have been empirically evaluated on real text corpora in comparison with two state of the art text clustering algorithms, bisecting k-means [4] and k-secting k-means [7]. The experiments were performed on a Pentium III PC with 400 MHz clock speed and 256 MB of main memory. We chose java 1.2 as the programming language because it allows fast and flexible development. For a fair comparison, we implemented all algorithms, FTC, HFTC and the k-means variants ourselves and used identical classes whenever possible. For generating the frequent term sets, we used a public domain implementation of the basic Apriori algorithm [11]. In section 4.1, we describe the test data sets. Sections 4.2 and 4.3 report the main experimental results for FTC and HFTC, respectively.

4.1 Data Sets

To test and compare cluster algorithms pre-classified sets of documents are needed. We used three different data sets which are widely used in other publications and reflect the conditions in a broad range of real life applications. These data sets are:

- *Classic*: Classic3 containes 1400 CRANFIELD documents from aeronautical system papers, 1033 MEDLINE documents from medical journals and 1460 CISI documents from information retrieval papers. We obtained the data set from ftp://ftp.cs.cornell.edu/pub/smart.
- *Reuters*: This test data set consists of 21578 articles from the Reuters news service in the year 87 (http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html). Typically, only a subset of 8654 articles, uniquely assigned to exactly one of these classes, is used for evaluation and we follow this procedure.
- *WAP*: During the WebACE project [12], 1560 web pages from the Yahoo! subject hierarchy were collected and classified into 20 different classes. We obtained this dataset from one of the authors, George Karypis.

A summary description of these data sets is given in Table 1.

Table 1. Test Data Sets

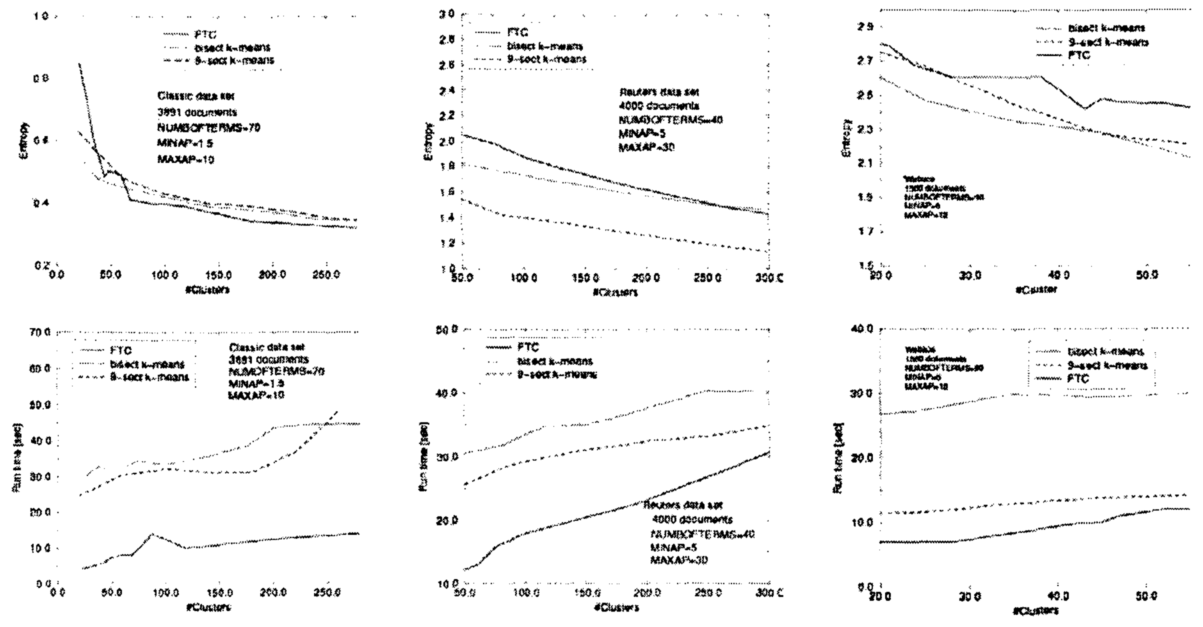
Data Set	# Documents	# Classes
Classic	3891	3
Reuters	8654	52
WAP	1560	20

4.2 Evaluation of FTC

To evaluate the quality of a flat, un-nested clustering we adopted the commonly used entropy measure. Let C be some clustering, i.e. a set of clusters, and $K = \{K_1, \ldots, K_k\}$ be the set of the true classes of a database. The *entropy* of the clustering C, denoted by $E(C)$, is defined as

$$E(C) = \sum_{C_j} \frac{n_j}{|D|} \sum_i - p_{ij} \ln(p_{ij})$$

where n_j means the size of cluster C_j and p_{ij} the probability that a member of Cluster C_j belongs to class K_i. The entropy measures the pureness of the clusters with respect to the classes and returns values in the interval [0...ln(|K|)]. The smaller the entropy, the purer are the produced clusters.



(a) *Classic* (b) *Reuters* (c) *WAP*

Figure 4. Comparison on the (a) *Classic*, (b) *Reuters* and (c) *WAP* data sets

Figure 4 compares the entropy as well as the runtime for FTC, bisecting k-means and 9-secting k-means for all three different data sets. For the experiments on the *Reuters* data set, we used a random sample of 4000 documents in order to speed-up the experiments. In addition to the standard preprocessing (see section 2), we performed a further step of term selection that is of advantage for FTC as well as for the k-means type algorithms: from the remaining terms in the corpus, we select the `numbofterms` ones with the highest variance of their frequency over all documents of the database. For a fair comparison, we had to consider the effect that the entropy measure favors larger numbers of clusters. Therefore, we measured clustering quality and runtime with respect to the number of clusters, i.e. we always compare clusterings with the same number of clusters for the different algorithms. Recall that the number of clusters is not a parameter of FTC, but depends on the minsup parameter.

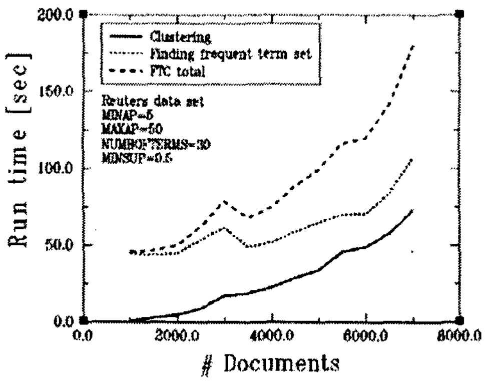


Figure 5: Scalability of FTC w.r.t. number of documents

In all experiments FTC yields a cluster quality comparable to that of bisecting and 9-secting k-means. However, FTC is significantly more efficient than its competitors on all test data sets. For example, FTC outperforms the best competitor by a factor of 6 on the Classic data (for 22 clusters), by a factor of 2 for the Reuters data (for 50 clusters) and by a factor of almost 2 (and factor 4 compared to bisecting k-means) for the WAP data (for 20 clusters). Furthermore, FTC automatically generates a description for the generated clusters while for the k-means type algorithms this has to be done in a extra postprocessing step.

We performed a set of experiments to analyze the scalability of FTC with respect to the database size. Figure 5 depicts the runtime of FTC with respect to the number of documents on the *Reuters* database. Even for the largest number of 7000 documents the runtime is still in the order of a few minutes which is a large improvement compared to other methods.

4.3 Evaluation of HFTC

To evaluate the clustering quality for hierarchical clustering algorithms, typically the F-Measure is used [7], [4]. We treat each cluster as if it were the result of a query and each class as if it were the relevant set of documents for a query. We then calculate the recall and precision of that cluster for each given class. More specifically, for cluster C_j and class K_i, the recall R and precision P are defined as: $R(K_i, C_j) = n_{i,j} / |K_i|$, and $P(K_i, C_j) = n_{i,j} / |C_j|$, respectively, where $n_{i,j}$ is the number of members of class K_i in cluster C_j. The *F-Measure* of cluster C_j and class K_i, denoted by $F(K_i, C_j)$, is then defined as

$$F(K_i,C_j)=\frac{2*R(K_i,C_j)*P(K_i,C_j)}{R(K_i,C_j)+P(K_i,C_j)}$$

For a hierarchical clustering C, the F-Measure of any class is the maximum value it attains at any node in the tree and an overall value for the F-Measure is computed by taking the weighted average of all values for the F-Measure as follows:

$$F(C)=\sum_{Ki\in K}\frac{|K_i|}{|D|}\max_{C_j\in C}\{F(K_i,C_j)\}\cdot$$

The F-Measure values are in the interval [0..1] and larger F-Measure values indicate higher clustering quality. We compared HFTC with the hierarchical versions of bisecting k-means and 9-secting k-means. The hierarchical algorithms do not have the number of clusters as a parameter, and the F-Measure does not have a similar bias as the entropy. Therefore, we obtain a single clustering of each test database for each of the competitors. Table 2 reports the resulting F-Measure values.

Table 2. Comparison of the F-Measure

Data Set	HFTC	bisecting k-means	9-secting k-means
Classic	0.50	0.81	0.56
Reuters	0.49	0.57	0.43
WAP	0.35	0.43	0.35

We observe that HFTC achieves similar values of the F-Measure as 9-secting k-means. Bisecting k-means yields significantly better F-Measure values than HFTC on all data sets. Note that the F-Measure forms some average of the precision and recall which favors non-overlapping clusterings. The two variants of k-means generate such non-overlapping clusterings, but HFTC discovers overlapping clusterings. However, overlapping clusters occur naturally in many applications such as in the Yahoo! directory.

5. CONCLUSIONS

In this paper, we presented a novel approach for text clustering. We introduced the algorithms FTC and HFTC for flat and hierarchical frequent term-based text clustering.

Our experimental evaluation on real text and hypertext data sets demonstrated that FTC yields a cluster quality comparable to that of state-of-the-art text clustering algorithms. However, FTC was significantly more efficient than its competitors on all test data sets. Furthermore, FTC automatically generates a natural description for the generated clusters by their frequent term sets. HFTC generates hierarchical clusterings which are easier to browse and more comprehendible than hierarchies discovered by the comparison partners.

Finally, we would like to outline a few directions for future research. We already mentioned that the integration of a more advanced algorithm for the generation of frequent term sets could significantly speed-up FTC and HFTC. FTC is a greedy algorithm. Other paradigms such as dynamic programming might also be adopted to solve the frequent term-based clustering problem and should be explored. Hierachical clusterings are of special interest for many applications. However, the well-known measures of hierarchical clustering quality do not adequately capture the quality from a user's perspective. New methods should be developed for this purpose. The proposed clustering algorithms have promising applications such as a front end of a web search engine. Due to the similarity of text data and transaction data, our methods can also be used on transaction data, e.g. for market segmentation. We plan to investigate these applications in the future.

6. REFERENCES

[1] Chakrabarti S.: Data mining for hypertext: A tutorial survey, *ACM SIGKDD Explorations*, 2000, pp.1-11.

[2] Cutting D.R., Karger D.R., Pedersen J.O., Tukey J.W.: Scatter / Gather: A Cluster-based Approach to Browsing Large Document Collection, *Proc. ACM SIGIR 92*, 1992, pp.318-329.

[3] Zamir O., Etzioni O.: Web Document Clustering: A Feasability Demonstration, *Proc. ACM SIGIR 98*, 1998, pp. 46-54.

[4] Steinbach M., Karypis G., Kumar V.: A Comparison of Document Clustering Techniques, *Proc. TextMining Workshop*, KDD 2000, 2000.

[5] Agrawal, R., Srikant R.: Fast Algorithms for Mining Association Rules in Large Databases, *Proc. VLDB 94*, Santiago de Chile, Chile, 1994, pp. 487-499.

[6] Kaufman L., Rousseeuw P.J.: *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 1990.

[7] Larsen B., Aone Ch.: Fast and Effective Text Mining Using Linear-time Document Clustering, *Proc. KDD 99*, 1999, pp. 16-22.

[8] Hipp J., Guntzer U., Nakhaeizadeh G.: Algorithms for Association Rule Mining – a General Survey and Comparison, *ACM SIGKDD Explorations*, Vol.2, 2000, pp. 58-64.

[9] Liu B., Hsu W., Ma Y.: Integrating Classification and Association Rule Mining, *Proc. KDD 98*, pp. 80-86.

[10] Zaiane O., Antonie M.-L.: Classifying Text Documents by Associating Terms with Text Categories, *Proc. Australasian Database Conference*, 2002, pp. 215-222.

[11] Yibin S.: An implementation of the Apriori algorithm, http://www.cs.uregina.ca/~dbd/cs831/notes/itemsets/dic.java, 2000.

[12] Han E.-H., Boly D., Gini M., Gross R., Hastings K., Karypis G., Kumar V., Mobasher B., and Moore J., WebACE: A Web Agent for Document Categorization and Exploration, Proc. Agents 98