Document clustering using character N-grams: a comparative evaluation with term-based and word-based clustering

Yingbo Miao
Vlado Keselj
Evangelos Milios

Technical Report CS-2005-23

September 18, 2005

Faculty of Computer Science
6050 University Ave., Halifax, Nova Scotia, B3H 1W5, Canada

Document clustering using character N-grams: A comparative evaluation with term-based and word-based clustering

Yingbo Miao, Vlado Keselj, Evangelos E. Milios

Contents

List of Tables **vi**

List of Figures **vii**

Abstract **viii**

1 Introduction **1**

2 Related Work **3**

2.1 Clustering 3

2.2 Document Clustering 5

2.3 Evaluation Measures 6

2.4 Automatic Term Extraction based on C/NC Value 7

2.5 N-gram application in text mining 8

3 Methodology and Implementation **10**

3.1 Document Clustering using N-grams 10

3.2 Document Clustering using Terms 10

3.3 Document Clustering using Words 11

3.4 K-Means Implementation 11

4 Experimental Results **14**

4.1 Data Set and Clustering Method 14

4.2 Zipf's Law 15

4.3 Experimental Results 16

4.3.1 Document clustering with N-grams 17

4.3.2 Feature selection based on document frequency 19

4.3.3 Comparing document clustering using N-grams, terms and words in different dime

5 Conclusion and Future work **34**

5.1 Conclusion 34

5.2 Future work 34

Bibliography **36**

List of Tables

2.1 Meanings of ?, + and $*$ in linguistic filters for $C/NCValue$ 8

2.2 Examples of byte, character and word N-grams 9

4.1 The most frequent Tri-grams, Terms and Words from the Reuters data set 15

4.2 The most frequent Tri-grams, Terms and Words from the cisi-cran-med data set 15

4.3 Clustering performance w.r.t. α 18

4.4 Clustering performance w.r.t. character N-gram size on Reuters data set 18

4.5 Clustering performance w.r.t. character N-gram size on cisi-cran-med data set(The N-gra

4.6 Clustering performance w.r.t. byte N-grams size on Reuters data set . 19

4.7 Clustering performance w.r.t. byte N-grams size on cisi-cran-med data set 19

4.8 Comparing clustering performance measured by entropy using Tri-grams, Terms and Wor

4.9 Comparing clustering performance measured by accuracy using Tri-grams, Terms and Wo

4.10 Comparing clustering performance measured by entropy using Tri-grams, Terms and Wor

4.11 Comparing clustering performance measured by accuracy using Tri-grams, Terms and Wo

4.12 The p-value of t-test results for clustering performance measured by entropy with Tri-gra

4.13 The p-value of t-test results for clustering performance measured by entropy with Tri-gra

4.14 The p-value of t-test results for clustering performance measured by entropy with Terms

4.15 The p-value of t-test results for clustering performance measured by entropy with Terms

4.16 Clustering performance measured by entropy with character Tri-gram, term and word rep

4.17 Clustering performance measured by accuracy with character Tri-gram, term and word re

List of Figures

4.1 Distribution of unique words on a log-log scale (Zipf's Law) 16
4.2 Distribution of N-grams on a log-log scale (Zipf's Law) 17
4.3 Clustering performance measured by Entropy and Accuracy w.r.t number of clusters on F
4.4 Running time per iteration of K-Means with Tri-grams on Reuters data set 20
4.5 Clustering performance using Tri-gram representation measured by entropy as a function
4.6 Clustering performance using Tri-gram representation measured by accuracy as a functio
4.7 Dimensionality of feature space as a function of minimum document frequency for feature
4.8 Clustering performance using Tri-gram representation measured by entropy as a function
4.9 Clustering performance using Tri-gram representation measured by accuracy as a functio
4.10 Dimensionality of feature space as a function of minimum document frequency for feature
4.11 Clustering performance using term representation measured by entropy and accuracy as a
4.12 Dimensionality of feature space as a function of minimum document frequency for feature
4.13 Clustering performance using term representation measured by entropy and accuracy as a
4.14 Dimensionality of feature space as a function of minimum document frequency for feature
4.15 Clustering performance using word representation measured by entropy and accuracy as a
4.16 Dimensionality of feature space as a function of minimum document frequency for feature
4.17 Clustering performance using word representation measured by entropy and accuracy as a
4.18 Dimensionality of feature space as a function of minimum document frequency for feature

Abstract

We propose a new method of document clustering with character N-grams. Traditionally, in *vector-space model*, each dimension corresponds to a word, with an associated weight equal to a word or term frequency measure (e.g. TFIDF). In our method, N-grams are used to define the dimensions, with weights equal to the normalized N-gram frequencies. We further introduce a new measure of the distance between two vectors based on the N-gram representation. In addition, we compare N-gram representation with a representation based on automatically extracted terms. Entropy and accuracy are our evaluation methods. T-test is used to prove there is significant differences between two results. From our experimental results, we find document clustering using character N-grams produces the best results.

Chapter 1

Introduction

Information retrieval through inverted indexing of document collection is widely used to retrieve documents from large electronic document collections, including the Word Wide Web. Conventionally, the results are represented as a list of documents, ordered by relevance. An alternative way of presenting the results is organizing them into clusters. Document clustering is an effective way to help people discover the required documents [16].

Clustering is the process of grouping a set of objects in to classes of similar objects [10]. A kind of traditional clustering methods is K-Means [18] and its variations [25]. For document clustering, the *Vector-space model* [24] can be used. In the *vector-space model*, each document is represented as a vector, where vector components represent certain feature weights. Traditionally, components of vectors are unique words. However, it has the challenge of high dimensionality. For the 8654 documents used in this thesis, which is a subset of Reuters-21578 [17], there are 32,769 unique words after stopword elimination and stemming. After removing numeral, there are still 18,276 words left. Because of the large and sparse document matrix, the distance between two documents approximately to be constant [9], which leads worse clustering quality. The high dimensionality also makes document clustering tasks take long time to get clustering results. Selection of words based on ad frequency criterion is commonly used to solve the high dimensionality problem [2]. We can also choose words based on their document frequency. The words appearing in too few or too many documents are not as important as other words and can be removed to reduce dimensionality.

Another approach to is using N-gram or term representation. Recent research shows that N-grams can be used for computer-assisted authorship attribution based on N-gram profiles [14]. Normalized N-gram frequencies are used to build author profiles with a new distance measure. Since the N-grams can be used in document classification and give good results, it is possible that N-gram is a better representation than word. We used a similar way to build a *vector space* for the documents and revised the distance measure to make it is suitable for document clustering. We performed experiments

In addition, we compare N-gram representation with a representation based on automatic term extraction. It is based on the idea that terms should contain more semantic information than words. Recent research shows the automatically extracted terms based on *C/NC Value* can be used as vector features and may lead to better clustering performance with lower dimensionality than word representation [20, 28]. However, the experiments in [20, 28] were performed on narrow domain text corpora (computer science, medical and web pages crawled from the .GOV Web site). We are interested in clustering performance of term representation on more generic data set, such as newswire stories.

In summary, this thesis addresses the following questions:

- Document clustering using N-grams: Is it meaningful to use N-grams as features?
- Document clustering using Terms: Is document clustering using automatic term extraction based on C/NC Value better on a generic text corpus than using unique words?
- How does feature selection based on document frequency compare with feature selection based on frequency on whole corpus?
- How do document clustering with N-grams, terms and words in different dimensionality compare?

The organization of the rest of the thesis is: Chapter 2 gives a brief review of document clustering, automatic term extraction with C/NC value and N-grams.Chapter 3 describes the details of document clustering with N-gram and term representation and the implementation of clustering method. Chapter 4 shows the experimental results and Chapter 5 summarizes the paper and describes the future work.

Chapter 2

Related Work

In this Chapter, some necessary background knowledge will be introduced. First we define what is clustering and introduce some existing clustering methods. We mainly describe the K-Means clustering algorithm and some of its variations since the K-Means algorithm is the base clustering algorithm we used. Second we describe the specific tasks for document clustering including document preprocessing, *vector-space model*, distance measures and evaluation measures. Third we describe the automatic term extraction based on *C/NC value*. At last of the Chapter we introduce N-grams and the applications of N-grams on text mining.

2.1 Clustering

Clustering is the process of grouping a set of objects into sets of similar objects. A cluster is a collection of data objects which are similar to one another within the same cluster and dissimilar to the objects in other clusters [10]. From the definition we see that the distance measure is essential to clustering. The details of choosing distance measure is described in 2.2.

Clustering analysis helps us discover overall distribution patterns and relationship among data attributes. Clustering is applied widely in many areas, including pattern recognition, data analysis, image processing, market research. Clustering can also be used in presenting search results [16]. It is more efficient for a user to locate relevant information among the retrieved documents than the traditional ranked list.

There exist many clustering algorithms, which can be classified into several categories, including partitioning methods, hierarchical methods and density-based methods. A partitioning method classifies objects into several one-level partitions. Each partition should contain at least one object. If each object belongs to only one cluster, it is called hard clustering; otherwise, it is call soft clustering. On the other hand, hierarchical methods create hierarchical decomposition of objects (e.g., a tree).

Two approaches for building hierarchy are bottom-up and top-down. The bottom-up approach, also called agglomerative approach, merges objects into small groups and small groups into larger groups until only one group is left. The top-down or divisive approach, splits whole data set into several groups. Then a large group is iteratively split up into smaller groups till every object is in only one group. A density-based method introduced the notion of density, the number of objects in the "neighborhood" of a object. A given cluster continues growing until its density exceeds a threshold. Density-based methods can build clusters of arbitrary shape and filter out outliers as noise.

One of the widely used clustering methods is K-Means [18], a partitioning clustering method. Given a cluster number k, K-Means randomly selects k positions in the same vector space as the objet representation as clusters centroids. K-Means assigns all objects to their nearest cluster, based on the distance between the objects and cluster centroids. Then the K-Means algorithms computes new centroids and adjusts clusters iteratively until all objects are converged or the maximum number of iterations is reached. The centroid can be mean point if the data objects are presented as vectors.

K-Means is a relatively scalable efficient clustering algorithm. Let n be the number of objects, k be the number of clusters and t be the iteration times, the computational complexity of K-Means is $\mathcal{O}(n \cdot k \cdot t)$ [10]. Normally, k and t are much smaller than n, so the running time is linear with the number of data objects. On the other hand, K-Means has some weaknesses. It cannot handle noisy data and outliers. As well, the initial points are chosen randomly, so a poor choice of initial points may lead to a poor result. Moreover, k, the number of clusters, should be specified in advance. Selecting the best k is often *ad-hoc* [8].

A lot of research has been performed on improving K-Means. Instead of using random initial points, we can give a good guess of starting points from clustering of a small subset of the data [3]. Multiple subsamples, say J, are chosen and clustered producing $J \times k$ centers. Then K-Means is applied to these centers J times to get the best mean as the initial points for whole data set. It is claimed that using this method can lead to better clustering results as well as reduce iterating times [3]. Another extension of K-Means is Bisecting K-Means [25], which is a top-down hierarchical clustering method using basic K-Means. At first, Bisecting K-Means groups all objects in a single cluster. Then it iteratively splits the largest cluster into two clusters using the basic K-Means until it reaches the desired number of clusters. Since the largest cluster is split in every iteration, Bisecting K-Means tends to produce clusters of similar size, which may not always be appropriate. We can use other ways to choose which cluster should be split, such as picking the one with the least overall similarity, or use a criterion based on both size and overall similarity. However, different ways of splitting have only slight effects on clustering results [25].

Density Based Spatial Clustering of Applications with Noise (DBSCAN) [6], is a density-based clustering method. Density of an object is based on how many other objects exist around it. DBSCAN treats a cluster as a maximal set of density-connected objects, whose density should be under a fixed threshold value given in advance. An object not in any cluster is considered as a noise. Given a radius value ϵ and a number of minimal points that should occur in around a dense object **MinPts**, DBSCAN scans through data set only one time, grouping neighboring objects into clusters. One of the advantages of DBSCAN is the ability of discovering clusters of arbitrary shape clusters as well as noise. In addition, unlike K-Means, DBSCAN can find out the cluster number automatically. Moreover, if R*-tree [21] is used, DBSCAN is efficient with run time complexity $O(n \times \log n)$. However, it is hard to tune DBSCAN's ϵ and **MinPts**. A bad pair of ϵ and **MinPts** may produce a single, giant cluster. RDBC(Recursive Density Based Clustering) algorithm [27] improves DBSCAN by calling DBSCAN recursively and adaptively changing its parameters. Comparing with DBSCAN, RDBC has same runtime complexity and is reported to give superior clustering results [27].

2.2 Document Clustering

Before clustering documents, several preprocessing steps may be needed, depending on the document representation chosen and the nature of the documents. First, HTML, XML or SGML tags should be removed. Second step is stopword elimination. Stopwords, such as "to", "of", and "is", are very common words. At last, word stemming is applied on corpus to get the stem of a word by removing the word prefix or suffix. Porter's stemming algorithm [22] is a widely used stemming algorithm for English language. The last two steps are required for word representation. In N-gram representation, the stopword elimination and word stemming are not required from our experimental results.

Vector-space model [23] is widely used in document clustering. In the *Vector-space model*, each document is represented by a vector of weights of n "features" (words, terms or N-grams) :

$$d_i = (tf_1, tf_2, \ldots, tf_m),$$

where m is the number of features and tf_i is the weight of i_{th} feature. If there are n documents in total, the corpus is represented by a $n \times m$ matrix X.

The weight of features should be calculated to build the matrix X. One of the feature weighting schemes is *TFIDF*, which combines the term frequency and document frequency. Term frequency is the frequency of a feature in a certain document, while document frequency is the number of documents where the feature appears. TFIDF is based on the idea that if a feature appears many times in a document, the feature is important to this document and should have more weight. A feature that appears in many documents is not important since it is not very useful in distinguishing different documents. Hence, it should have lower weight. Let $tf(i,j)$ be the term frequency of feature j in a document d_i , $df(j)$ be the document frequency of feature j and N be the number of documents in the whole collection, *TFIDF* is defined as [24]:

$$TFIDF(i,j) = tf(i,j) \cdot \log\left(\frac{N}{df(j)}\right)$$

Distance or similarity is fundamental of defining a cluster. Euclidean distance is one of the most popular distance measures. Given two vectors $d_1 = (d_{1,1}, d_{1,2}, \ldots, d_{1,m})$ and $d_2 = (d_{2,1}, d_{2,2}, \ldots, d_{2,m})$, their Euclidean distance is [26]:

$$\sqrt{\sum_{i=1}^{m} (d_{1,i} - d_{2,i})^2},$$

which is one of the standard metrics for geometrical problems. The *cosine* measure, however, is more popular in document clustering, because it factors out differences in length between vectors [26]. Cosine measures the *similarity* between two documents d_1 and d_2 [26] as:

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

where "·" denotes the vector dot product and "‖" denotes the length of a vector. It is the *cosine* of the angle between two vectors in n-dimension space. Euclidean and cosine are equivalent if the vectors are normalized to unit length.

2.3 Evaluation Measures

There are two kinds of evaluation measures: *internal quality* measure and *external quality* measure [25]. An *internal quality* measure references none external knowledge. It is used if the class label of each document is unknown. On the other hand, if an *external quality* measure compares clustering results with given classes to measure the clustering quality. Some of *external quality* measures are: Entropy [2], F-measure [15] and Accuracy.

Let $C = \{c_1, c_2, \ldots, c_l\}$ be the set of clusters, $K = \{k_1, k_2, \ldots, k_m\}$ be the set of given classes and D be the set of documents, the entropy $E(C)$ is defined as

$$E(C) = \sum_{c_j} \frac{|c_j|}{|D|} \sum_i -p_{ij} \ln(p_{ij}), \tag{2.1}$$

where p_{ij} is the "probability" that a member of cluster c_j belongs to class k_i. The formal definition of p_{ij} is

$$p_{ij} = \frac{|c_j \cap k_i|}{|c_j|} \tag{2.2}$$

The range of entropy is $(0, \ln|K|)$[2]. Entropy measures the "purity" of the clusters with respect to the given classes. The smaller the entropy, the purer are the clusters.

Accuracy $A(C)$ is defined as:

$$A(C) = \frac{1}{|D|} \sum_{C_j} \max_i |c_j \cap k_i| \tag{2.3}$$

Accuracy shows the clustering results with respect to the main classes. The basic idea of Accuracy is try to find a "main " class of each cluster. The main class of a cluster is the class which gives the largest intersection with the cluster. The value of accuracy lies in $(\frac{\max_i |k_i|}{|D|}, 1)$. The larger the accuracy, the better is the result.

2.4 Automatic Term Extraction based on C/NC Value

The *C/NC Value*, a state-of-the-art method for automatic term extraction, combines both linguistic (linguistic filter and part-of-speech tagging [4]) and statistical information (frequency analysis and C/NC value) [7, 20]. The C_Value aims to improve the extraction of nested multi-word terms while the NC_Value aims to improve multi-word terms extraction in general by incorporating context information into C_Value method.

The linguistic part of C_Value computation includes part-of-speech (POS) tagging and a linguistic filter. The POS tagging program assigns to each word of a text a POS tag, such as Noun, Verb or Adjective. Then, one the following linguistic filters is used to extracted word sequence.

1. Noun$^+$Noun

2. (Adj|Noun)$^+$Noun

3. ((Adj|Noun)$^+$|((Adj|Noun)*(NounPrep)$^?$)(Adj|Noun)*)Noun

The meanings of "?", "+"and "∗" , as shown in Table 2.1 are same as their meanings in regular expressions.

Symbol	Meaning
?	Matches the preceding element zero or one times
+	Matches the preceding element one or more times
∗	Matches the preceding element zero or more times

Table 2.1: Meanings of ?, + and ∗ in linguistic filters for $C/NCValue$

The first filter is "closed filter" and gives higher precision but lower recall. The last two filters are "open filters" and produce higher recall but lower precision.

After POS tagging and linguistic filtering, we calculate the C_Value:

$$C\_\text{value}(\alpha) = \begin{cases} \log_2 |\alpha| f(\alpha) & \alpha \text{is not nested} \\ \log_2 |\alpha| (f(\alpha) - \frac{1}{P(T_\alpha)} \sum_{\beta \in T_\alpha} f(\beta)) & \text{otherwise} \end{cases}$$

where α is the candidate string, $|\alpha|$ is the number of words in string α, $f(\alpha)$ is the frequency of occurrence of α in the corpus, T_α is the set of extracted candidate terms that contain a, and $P(T_\alpha)$ is the number of these terms which contain α. Nested terms are terms which appear within other longer terms.

The NC_Value extended C_Value by using context word information into term extraction. Context words are those that appear in the vicinity of candidate terms, i.e. nouns, verbs and adjectives that either precede or follow the candidate term.

Automatic term extraction based on *C/NC Value* performs well on domain specific document sets [7, 20]. The terms can be used as features in document vectors. Using term representation has the potential of reducing significantly reduce the dimensionality and giving better results than word representation in special text corpora [28]. Recent research on parallel term extraction makes it possible to perform automatic term extraction on large(Gigabyte-sized) text corpora [29] .

2.5 N-gram application in text mining

An N-gram is a sequence of symbols extracted from a long string [5]. The symbol can be a byte, character or word. Extracting character N-grams from a document is like moving a n character wide "window" across the document character by character.

Each window position covers n characters, defining a single N-gram. In this process, any non-letter character is replaced by a space and two or more consecutive spaces are treated as a single one. The byte N-grams are N-grams retrieved from the sequence of the raw bytes as they appear in data files, without any kind of preprocessing. Table 2.2 shows examples of byte, character and word Bi-grams extracted from the string "byte n-grams".

	Samples
Byte Bi-grams	_n -g am by e_ gr m\n n- ra te yt
Character Bi-grams	_G _N AM BY E_ GR M_ N_ RA TE YT
Words Bi-grams	byte_n n_gram

Table 2.2: Examples of byte, character and word N-grams

A N-gram representation was used in authorship attribution [14]. The author profile is generated from training data as a set of L pairs $\{(x_1, f_1), (x_2, f_2), \ldots, (x_L, f_L)\}$, where x_j is a most frequent N-gram and f_j is the normalized frequency of x_j ($j = 1, 2, \ldots, L$). The N-gram frequency is normalized by dividing by the sum of all N-gram frequencies of the same N-gram size and in the same document. The distance between two profiles is defined as:

$$\sum_{n \in profile} \left(\frac{f_1(n) - f_2(n)}{\frac{f_1(n)+f_2(n)}{2}} \right)^2 \tag{2.4}$$

where $f_i(n) = 0$ if n is not in the profile.

Experiments of byte N-gram based authorship attribution shows that it perform well on English, Greece and Chinese corpora [14]. The accuracy on Chinese corpora is not as high as other corpora. The restricted profile size and the fact that Chinese characters use two bytes may be the reason. The advantages of N-gram based authorship attribution also include controllable profile size and a simple algorithm.

Chapter 3

Methodology and Implementation

In this chapter, we will introduce how N-gram representation is used in document clustering. Document clustering using terms and words are also describes. Finally, we described limitations of some existing K-Means algorithms, which is why we implemented K-Means algorithm.

3.1 Document Clustering using N-grams

In our approach, we attempted to follow the idea proposed in [14]. We retrieve the most frequent N-grams and their frequencies from each document as its profile. To measure the distance between two documents, we revise Eq. (2.4) to Eq. (3.1):

$$\sum_{1 \leq j \leq L} \frac{(v_1(j) - v_2(j))^2}{\left(\frac{(v_1(j)+v_2(j))}{2}\right)^{\alpha}} \tag{3.1}$$

When $\alpha = 2$, Eq. (3.2) is same as Eq. (2.4) and when $\alpha = 0$, Eq. (3.2) is the square of *Euclidean distance*. However, from our experimental results, we find that $\alpha = 1$ produces the best result. So the distance formula is:

$$\sum_{1 \leq j \leq L} \frac{(v_1(j) - v_2(j))^2}{\frac{(v_1(j)+v_2(j))}{2}} = \sum_{1 \leq j \leq L} \frac{2 \cdot (v_1(j) - v_2(j))^2}{(v_1(j) + v_2(j))} \tag{3.2}$$

Algorithm 1 gives the algorithm of document clustering with N-grams. For each document, a profile is built using most frequent N-grams in the document. All unique N-grams in these profiles are used as vector components. Thus, the *vector space* can be built to be used in the K-Means algorithm.

3.2 Document Clustering using Terms

Terms are extracted automatically based on their *C/NC Value*. In order to reduce the dimensionality, m terms with highest *C/NC Value* are chosen as vector features.

Algorithm 1 Document Clustering using N-grams

V: Vocabulary of N-grams
for all document d_i **do**
  $V \leftarrow V \bigcup$ the most frequent N-grams of d_i
end for
M: The document Matrix, the columns of which are the document vectors
for all document d_i **do**
  Build Profile P_i as $\{(x_1, w_1), (x_2, w_2), \ldots, (x_L, w_L)\}$
  // x_i is the n-gram and w_i is the weight of x_i
  for all $e_j \in V$ **do**
    if $e_j \in P_i$, where $e_j == x_k$ **then**
      $M[i][j] = w_k$
    else
      $M[i][j] = 0$
    end if
  end for
end for
Feature selection on M based on document frequency
Clustering M using K-means with Formula 3.2 as distance formula.

TFIDF is used as the weighting scheme.

We can use *cosine* as similarity measure or *sine* as distance measure, which is claimed to be better than Euclidean distance in document clustering [26]. When calculate *cosine* or *sine* between two document vectors, each vector corresponds to a point on the unit sphere. So real distance between two vectors is the length of curve between the two corresponding points and it is *arcsin*. Hence, we use Eq. (3.3) to measure *distance*.

$$arcsin(d_1, d_2) = \frac{\pi}{180} \cdot \sqrt{1 - cosine(d_1, d_2)^2} \tag{3.3}$$

3.3 Document Clustering using Words

Stopword elimination and word stemming are required for word representation. Only the most frequent words are used to reduce dimensionality. The weighting scheme is TFIDF. The *arcsin* is used as distance measure

3.4 K-Means Implementation

We use K-Means as base clustering method in this project since K-Means is a widely used, relatively efficient clustering method. In addition, simple K-Means can be easily revised to Bisecting K-Means to generate hierarchical clusters.

Algorithm 2 shows the details of K-means algorithm [10]. We considered but did not use initial point refinement method described in [3] since, in our experiments, using the initial point refinement does not produce better clustering quality, nor smaller iteration times than randomly choosing initial points. Moreover, the experiment in [3] is built on 10 sub samples($J = 10$) and only one sub sample($J = 1$). When $J = 10$, each sample size is 10% of the whole data set. In this case the running time of looking for initial points is nearly the same as the running time of applying the simple K-Means on whole data set, since K-Means time-complexity is linear with the number of objects. When $J = 1$, the performance is not significantly better or it is it may be even worse than randomly choosing initial points on "high" dimensionality. For this reason we decided not to use this method.

Algorithm 2 K-means

```
Partition objects into k nonempty subsets randomly.
repeat
  Compute the centroids of the clusters.
  Assign each object to the cluster with the nearest centroid.
until No object is moved or reach the maximum number of iterations
```

The K-Means algorithm is implemented in several software packages, including Weka [11], Matlab and CLUTO [12]:

- **Weka**: Weka is a widely used Machine learning toolkit implemented in Java. Weka contains many machine learning tools for classification, regression, clustering and association rules. It is an open source software issued under GNU General Public License, so we can easily revise its K-Means implementation to use Eq. 3.2 as the distance measure. Weka also stores sparse data efficiently. Another advantage of Weka is that it implements several clustering methods besides K-Means, such as EM. That makes it easier to perform tests on different clustering methods. Scalability, however, is the main problem of Weka. For large data sets, it took several days to perform clustering. That is why we decided to use faster software to conduct our experiments.

- **Matlab**: A K-Means clustering implementation in Matlab is available [19]. It supports several distance measures including squared Euclidean distance and cosine. When we used cosine as distance measure, however, differences of distances were so small that Matlab gave out error messages and stopped. In addition, we found Matlab was even slower than Weka when we used the Eq. 3.2 as the distance measure.

- **CLUTO**: CLUTO is another software package for clustering. CLUTO claims that it can scale to the large data set containing hundreds of thousands of objects and tens of thousands of dimensions. Unfortunately, CLUTO is not an open source software. We cannot revise it to use Eq. 3.2 as distance measure.

Since none of these implementations satisfy our requirement of speed and modifiability, we implemented K-Means algorithm in C++ by ourselves.

Chapter 4

Experimental Results

4.1 Data Set and Clustering Method

We used Reuters-21578, Distribution 1.0 [17] and cisi-cran-med [1] data sets to test and compare our clustering algorithms. Reuters-21578 contains 21,578 Reuters newswire stories. After removing the articles having less or more than one topic, 8,654 documents are remain which belong to 65 topics. We took topics as class label, so there are 65 given classes. The cisi-cran-med data set contains the Information Retrieval abstract(cisi), Aeronautics abstracts(cran) and Medline abstracts(med). The cisi-cran-med data set has 3,891 documents.

Documents in the Reuters data set are not spread in the 65 given classes evenly. The two largest classes contain 5,860 documents and there are 23 classes have less then 10 documents. About 86.93% documents are in the 10 largest classes, while only about 0.64% documents are in the 20 smallest classes.

For N-gram and term representation, the documents are used without any preprocessing. For word representation, we removed stop-words and did word stemming using Porter's stemming algorithm [22].

K-Means is the base clustering method to compare the three representations. In order to decrease the random seed impact on K-Means, we ran each test 10 or 30 times and used the mean as the final result. Ten or thirty, however, is still a small number. So the t-test is used to prove one group of results is better than another group of results.

The ten most frequent character Tri-grams, terms and words retrieved from the Reuters and cisi-cran-med are shown in Table 4.1 and Table 4.2. These examples give us a general idea of retrieved N-grams, terms and words. The "_" in n-grams means the space character. The word "the" is so frequent that it produces the first three Tri-grams. In word representation, whereas, "the" is removed as a stopword. We can see that many extracted terms are not real terms.

Also automatic term extraction based on *C/NC Value* performs well in domain-specific corpora such as computer science and medical articles [20], but it seems to have a lower accuracy on the newswire articles. We will show it later that document clustering with terms, nevertheless, still gives better clustering performance than with words.

	Samples
Tri-grams	_TH THE HE_ _IN ER_ ED_ _OF _TO _RE _CO
Terms	mln dlr, mln vs, oil india ltd ,cts vs, oil india, india ltd, vs loss, dlrs vs, avg shr, mln avg shr
Words	said mln dlr ct vs year net compani pct shr

Table 4.1: The most frequent Tri-grams, Terms and Words from the Reuters data set

	Samples
Tri-grams	_TH THE HE_ _OF OF_ _IN ION ED_ TIO ON_
Terms	mach number, boundary layer,heat transfer, laminar boundary layer, pressure distribution, shock wave, growth hormone, laminar boundary, leading edge, turbulent boundary layer
Words	flow, result, number, effect, librari, pressur, inform, method, present, studi

Table 4.2: The most frequent Tri-grams, Terms and Words from the cisi-cran-med data set

4.2 Zipf's Law

Zipf's law [30] describes the distribution of words used in human generated texts. If we count how many times each word appears in the whole corpus, then we can sort words by their frequencies. We can give the most frequent word rank 1, the second most frequent word rank 2 and up to the least frequent word rank n if there are n unique words. Let r_i be the i_{th} word in the sorted list, f_i be the frequency of r_i and C is a constant, then the Zipf's law is:

$$r_i \times f_i = C, \tag{4.1}$$

where $i = 1, 2, \ldots, n$. If we take the *log* to each side of Equation (4.1), then we get:

$$\log r_i + \log f_i = C_2, \tag{4.2}$$

where $C_2 = \log C$ is a constant too. So if r_i and f_i are plotted on a log-log scale, we should get a straight line with gradient -1.

The distribution of unique words obeys Zipf's law, as shown in Fig. 4.1, in Reuters data set.The distribution of N-grams on Reuters data set, is shown in Fig. 4.2, for $1 \leq N \leq 10$. We can find that for $N \geq 3$, N-grams obey Zipf's law in a rough approximation. The gradients of lines are not -1, which implies that a more general rule must be used: $r_i^\beta \times f_i^\beta = C$, where β is a constant. Another fact we can deduce from Fig. 4.2 is that the curves representing larger N are always lower than curves representing smaller N. It means the larger the N, the less are there less common N-grams. That is why in Table 4.4, the sparsity of document matrix is increasing with increasing the N-gram size.

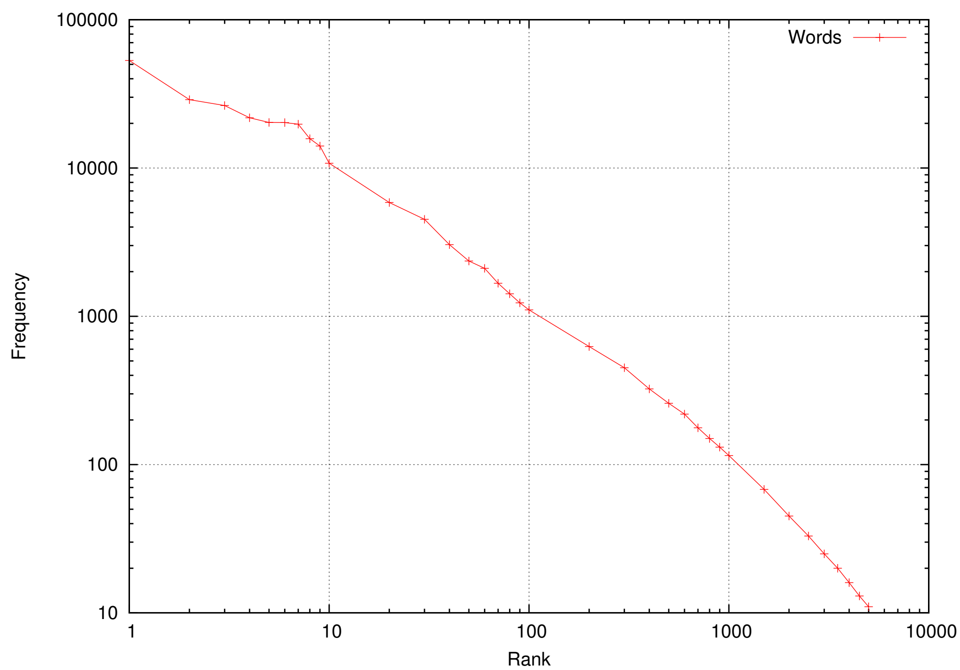

Figure 4.1: Distribution of unique words on a log-log scale (Zipf's Law)

4.3 Experimental Results

In this section, we show our experimental results of document clustering using N-grams, terms and words. For document clustering with N-gram representation, clustering results are influenced by:

- α in Eq. (3.1)
- N-gram size

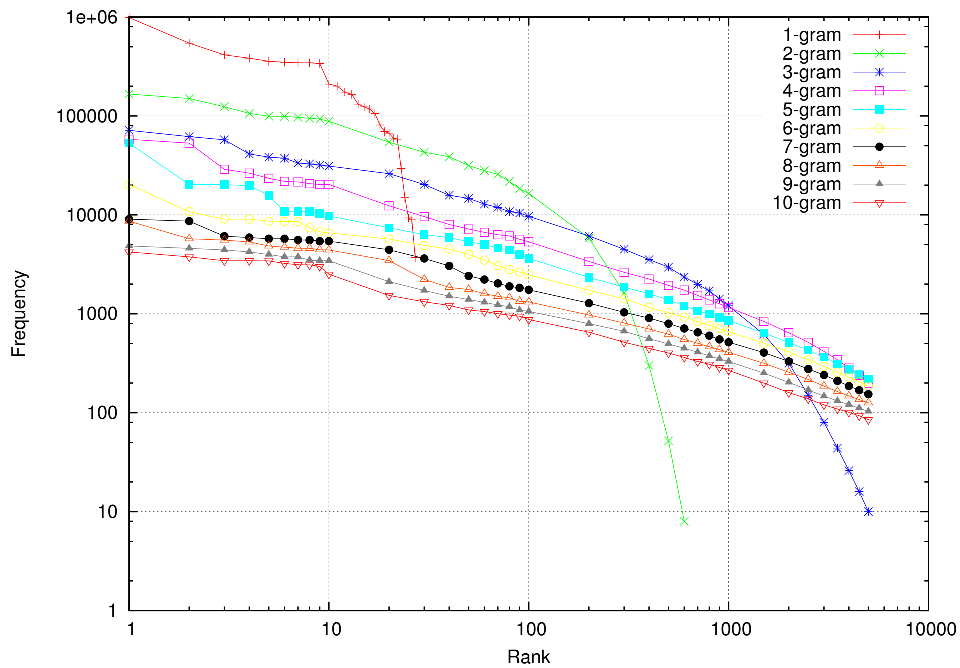

Figure 4.2: Distribution of N-grams on a log-log scale (Zipf's Law)

- Byte N-gram representation or character N-gram representation
- Number of clusters(k)

The experimental results for turning document clustering with N-grams are shown in 4.3.1. Furthermore, we performed experiments with feature selection based on document frequency for N-gram, term and word representation and show the results in 4.3.2. At last, we shows the results of comparing document clustering using N-grams, terms and words in different dimensionality in 4.3.3

4.3.1 Document clustering with N-grams

In Eq.(3.1), we introduced α. The best clustering results, as shown in Table 4.3,is deduced when $\alpha = 1$. The result of $\alpha = 2$ is not included in Table 4.3 since in this case all documents converge to a single cluster after iterating enough times. When $\alpha = 0$, Eq. (3.2) is same as the squared *Euclidean distance*, and gives worse results than $\alpha = 1$. That is why we use Eq. (3.2) as our distance measure.

N-grams size, as shown Table 4.4, has impacts on clustering quality as well as vector dimensionality. Due to the time and memory limitation, N-grams appearing in less than 0.2% documents are deleted for $n \geq 4$. Dimensionality and sparsity increase with increasing N-gram size. In other words, the larger the N-gram size, the less

α	Entropy	Accuracy
0	1.116	0.678
1	0.689	0.803

Table 4.3: Clustering performance w.r.t. α

are there the common N-grams among documents. Quad-gram representation gives the best entropy and accuracy on the Reuters data set, while on cisi-cran-med data set, the 5-ngrams gives the best clustering quality. Tri-gram representation, however produces comparable entropy and accuracy with a more practical dimensionality.

N	Dimensionality	Sparsity	Entropy	Accuracy
2	671	78.45%	0.821	0.774
3	7978	96.05%	0.689	0.803
4	12559	96.79%	0.645	0.814
5	24760	98.29%	0.683	0.803
6	32160	98.78%	0.924	0.790

Table 4.4: Clustering performance w.r.t. character N-gram size on Reuters data set

N	Dimensionality	Sparsity	Entropy	Accuracy
2	516	63.16%	0.382	0.890
3	4066	89.35%	0.368	0.890
4	14968	96.22%	0.327	0.905
5	32418	98.11%	0.288	0.920
6	47940	98.75%	0.330	0.890

Table 4.5: Clustering performance w.r.t. character N-gram size on cisi-cran-med data set(The N-grams appearing in less than 8 documents are deleted)

Experiments using *byte* and *character* N-grams are performed. These two types of N-grams are adopted from the software tool Ngrams [13]. The *byte N-grams* are N-grams retrieved from the sequence of the raw bytes as they appear in data files, without any kind of preprocessing. The *character N-grams* are retrieved after translating all letters into their upper-case forms, and any sequence of non-letter bytes is replaced by a space.

Byte N-grams representation gives different results with Reuters data set and cisi-cran-med date set, as shown in Table 4.6 and Table 4.7. On Reuters data set, it produces better results than on cisi-cran-med data set. Generally, character N-grams produce better results than byte N-grams. Moreover, dimensionality and sparsity with character N-gram representation are lower than with byte N-gram representation. Our remaining experiments were performed on character Tri-grams.

N	Dimensionality	Sparsity	Entropy	Accuracy
2	1473	87.77%	0.869	0.761
3	16114	97.72%	0.705	0.801
4	17914	97.53%	0.657	0.811
5	29828	98.51%	0.752	0.796
6	34909	98.86%	0.752	0.778

Table 4.6: Clustering performance w.r.t. byte N-grams size on Reuters data set

N	Dimensionality	Sparsity	Entropy	Accuracy
2	1473	87.77%	1.069	0.418
3	16114	97.72%	1.077	0.398
4	17914	97.53%	1.079	0.391
5	29828	98.51%	1.073	0.393
6	34909	98.86%	1.030	0.422

Table 4.7: Clustering performance w.r.t. byte N-grams size on cisi-cran-med data set

In K-Means algorithm, the number of clusters, k, should be given in advance. Selecting the best k, however, is done by trial and error. Experiments using different cluster numbers (k) are performed and shown in Fig. 4.3. We see that result is becoming better when the number of clusters increases. The curve of entropy drops very quickly at first. However, after the number of clusters gets larger than 55, the curve drops slowly. The difference of entropies with 55 clusters and 85 clusters is only about 0.056. In other words, we can choose any k between 55 and 85. Our remaining experiments were performed with $k = 65$ for the Reuters data set and $k = 3$ for the cisi-cran-med data set.

4.3.2 Feature selection based on document frequency

Both character N-gram and byte N-gram representation produce high dimensionality and spars data. The high dimensionality does not only affect clustering quality, but also results in long running time. The running time per iteration, as shown in Fig. 4.4, increases linearly with increasing the dimensionality. So if we can reduce the dimensionality, we also decrease the running time effectively. This is one of the motivations of feature selection.

We performed experiments on feature selection based on document frequency. The experiments were performed with N-gram, term and word representation to see

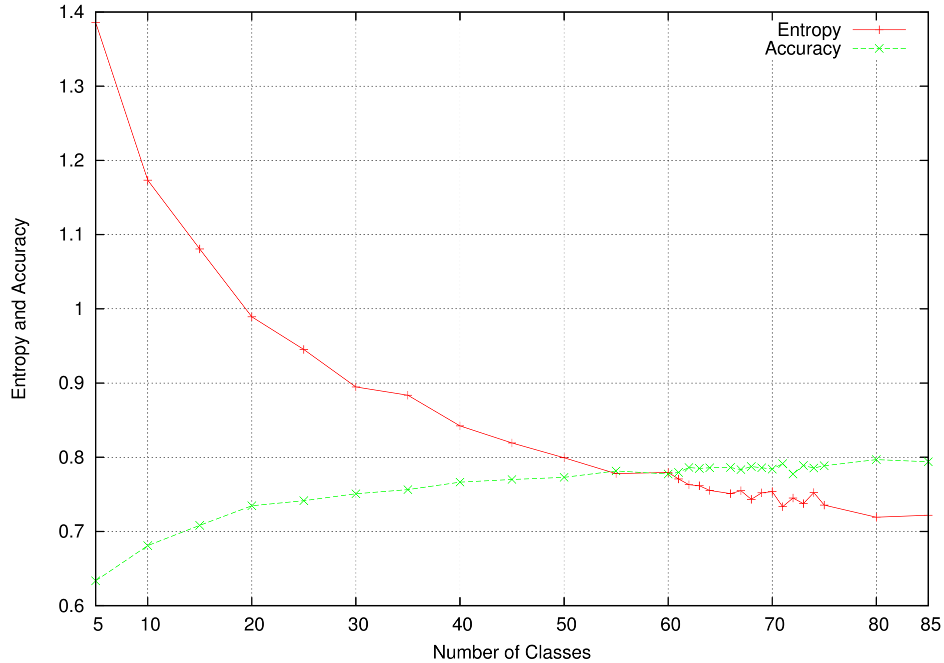

Figure 4.3: Clustering performance measured by Entropy and Accuracy w.r.t number of clusters on Reuters data set

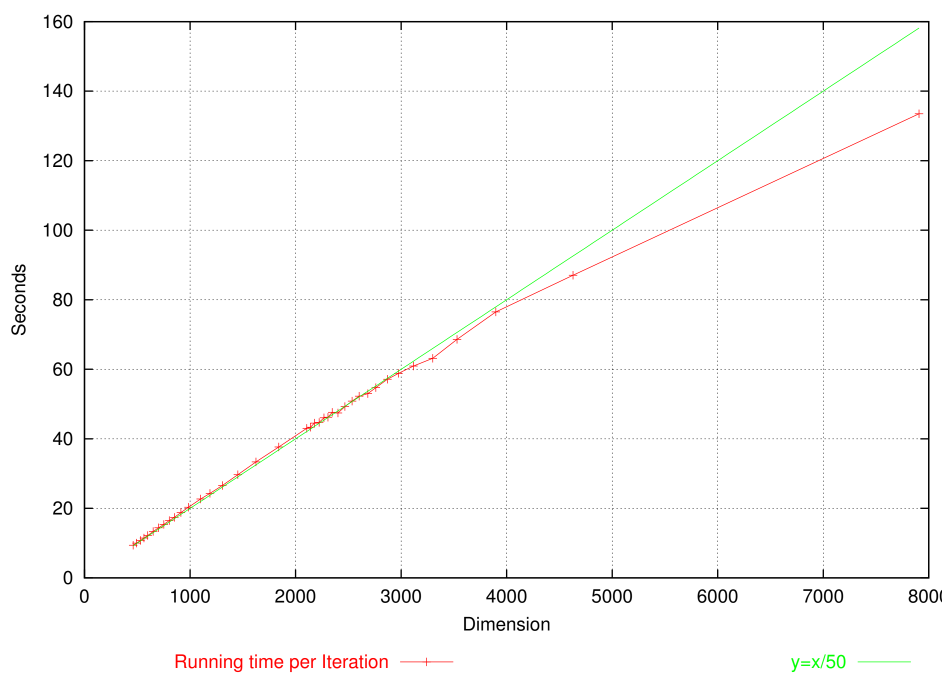

Figure 4.4: Running time per iteration of K-Means with Tri-grams on Reuters data set

how document frequency affects on clustering results and the differences for the three representations.

We changed two values of their document frequency: minimum document frequency df_{min} and maximum document frequency df_{max}. Minimum document frequency means for an N-gram to be included in the vector space definition, it should appear in at least df_{min} documents while maximum document frequency means it should appear in at most df_{max} documents.

The clustering quality becomes worse with the increasing of minimum document frequency on the Reuters data set, as shown in Fig. 4.5 and Fig. 4.6. For each curve in these two figures, the maximum *df* is fixed. The length of a error bar corresponds to the standard deviation of ten times running results in the certain point. However, there is only little difference when minimum *df* is between $(0, 270)$. For same minimum *df*, a lower maximum *df* produces better results when maximum *df* is in $(3000, 8654)$. The dimensionality, as shown in Fig. 4.7, shrinks very quickly with increasing minimum df. More than 7000 Tri-grams only appear in less than 800 documents. On the other hand, the maximum *df* has little impact on dimensionality. In fact, only 170 Tri-grams appears in more than 3,000 documents and 83 Tri-grams appears in more than 4,000 documents.

The cisi-cran-med data set has similar results, as shown on Fig. 4.8, Fig. 4.9 and Fig. 4.10. When increase the minimum document frequency, the quality of clustering results becomes better at first until the minimum document frequency is around 200, and becomes worse later. We had some informal tests on some small subsets of the Reuters data set, which give similar results. The results prove that the N-grams with low document frequency are not important and should be deleted as noises.

The same experiments for term and word representation are performed too. The results with term representation are shown in Fig. 4.11, Fig. 4.12, Fig. 4.13 and Fig. 4.14, while results with word representation are shown in Fig. 4.15, Fig. 4.16, Fig. 4.17 and Fig. 4.18. The length of a error bar corresponds to the standard deviation of ten times running results in the certain point. We see that when minimum *df* increases, results with term representation and word representation become worse quickly. As well, comparing with Tri-gram, there are fewer common terms or words. More than 7,000 terms (or words) appears only in less than 20 documents in the Reuters data set.

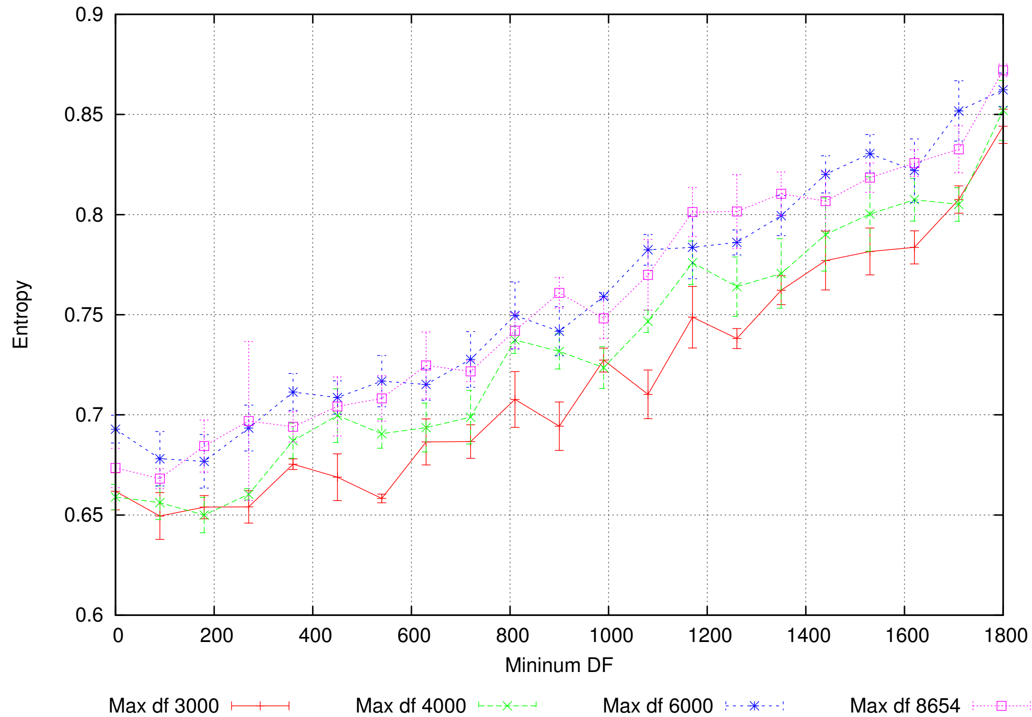

Figure 4.5: Clustering performance using Tri-gram representation measured by entropy as a function of the minimum document frequency for feature selection on the Reuters data set

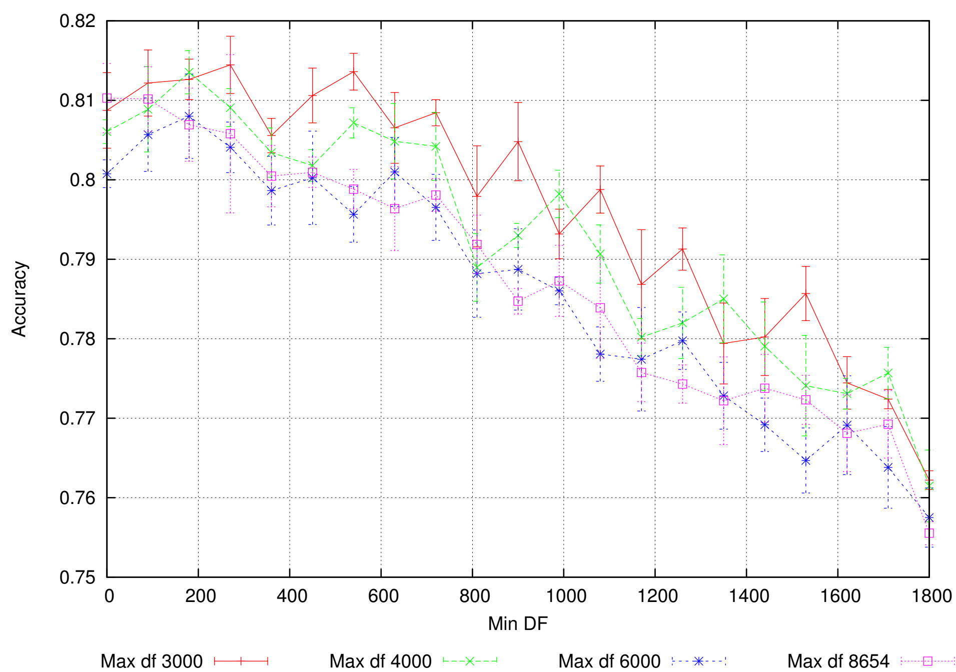

Figure 4.6: Clustering performance using Tri-gram representation measured by accuracy as a function of the minimum document frequency for feature selection on the Reuters data set

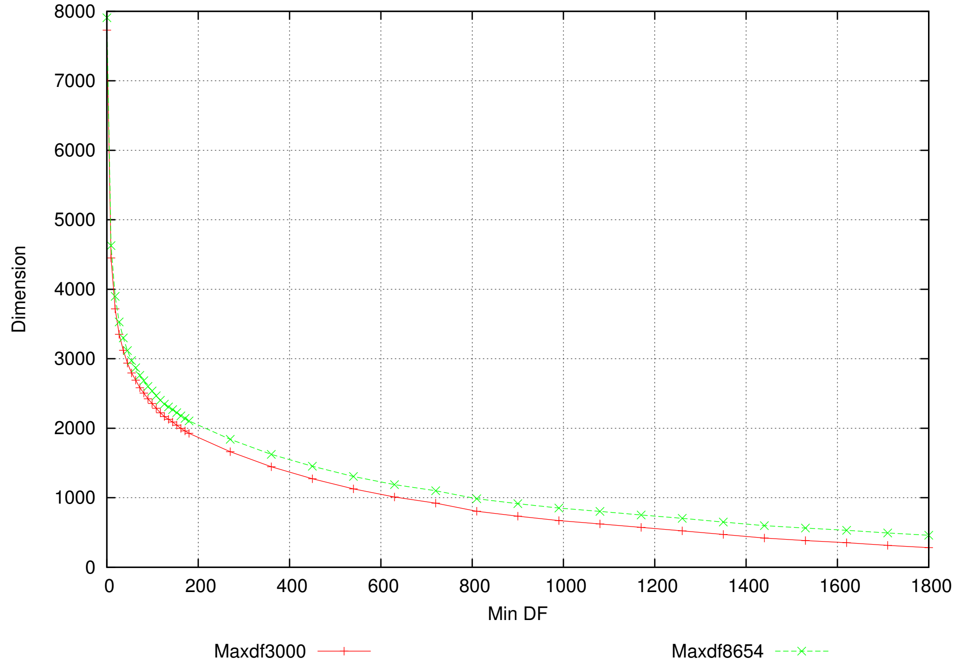

Figure 4.7: Dimensionality of feature space as a function of minimum document frequency for feature selection with Tri-gram representation on the Reuters data set

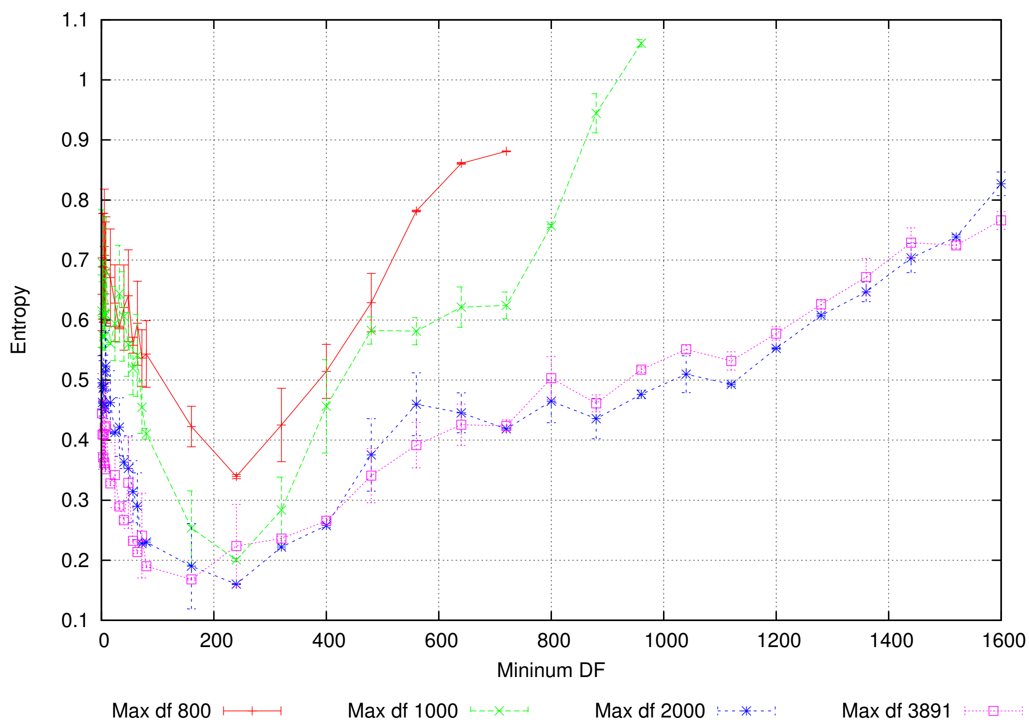

Figure 4.8: Clustering performance using Tri-gram representation measured by entropy as a function of the minimum document frequency for feature selection on the cisi-cran-med data set

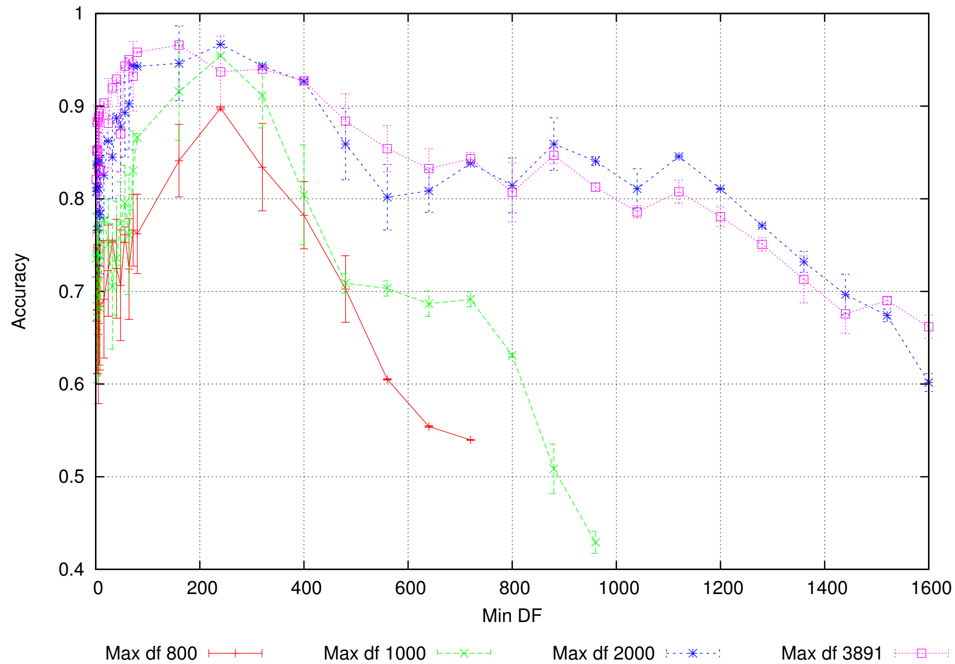

Figure 4.9: Clustering performance using Tri-gram representation measured by accuracy as a function of the minimum document frequency for feature selection on the cisi-cran-med data set

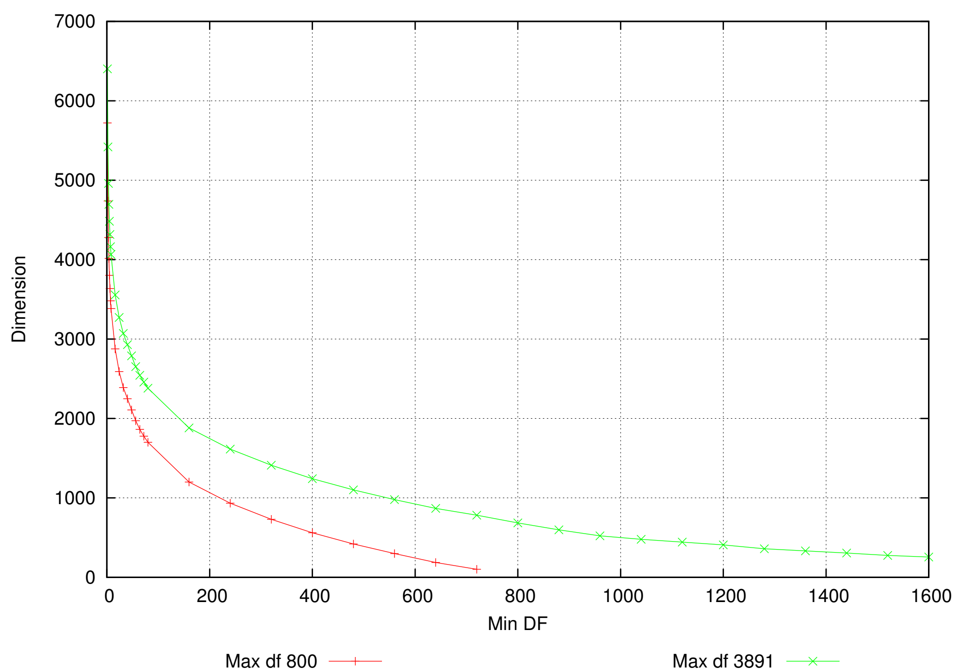

Figure 4.10: Dimensionality of feature space as a function of minimum document frequency for feature selection with Tri-gram representation on the cisi-cran-med data set

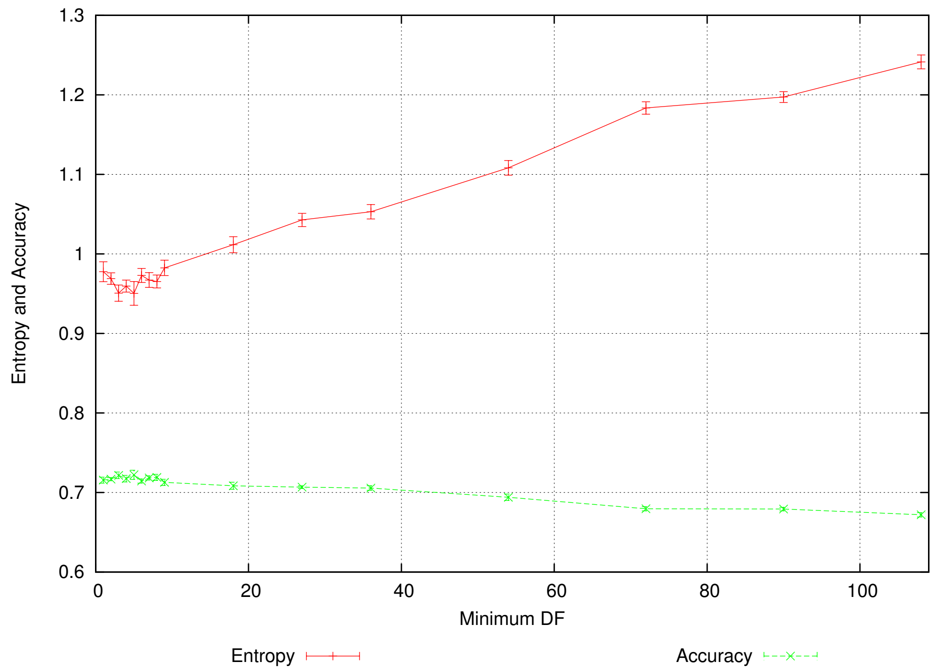

Figure 4.11: Clustering performance using term representation measured by entropy and accuracy as a function of the minimum document frequency for feature selection on the Reuters data set

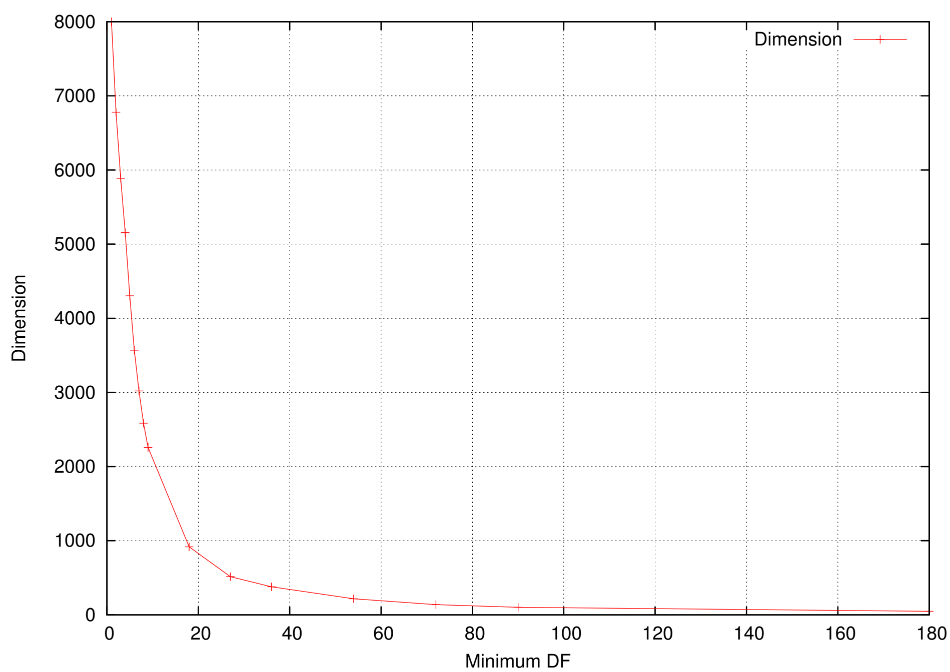

Figure 4.12: Dimensionality of feature space as a function of minimum document frequency for feature selection with term representation on the Reuters data set

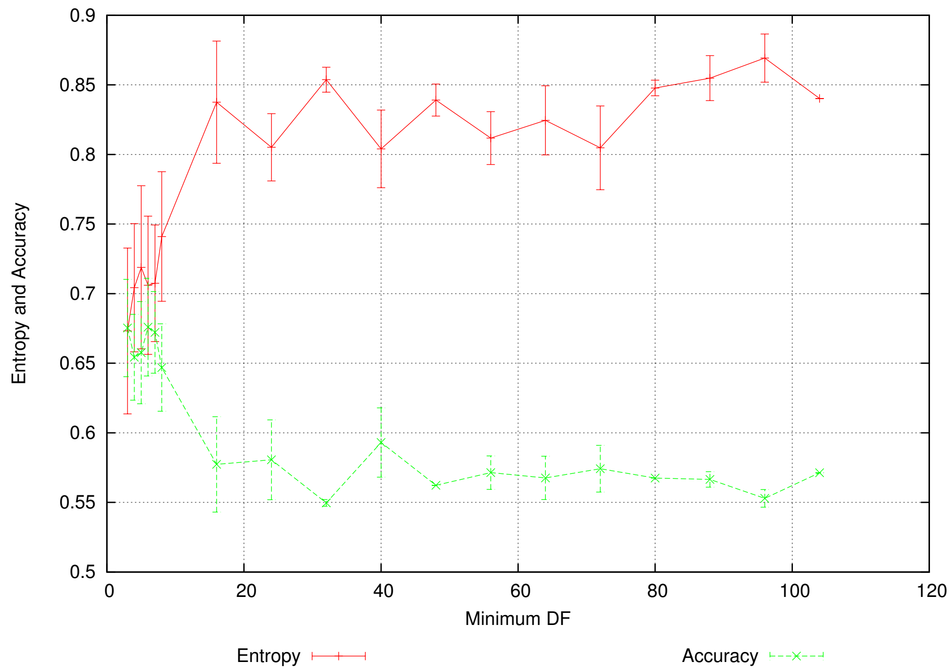

Figure 4.13: Clustering performance using term representation measured by entropy and accuracy as a function of the minimum document frequency for feature selection on the cisi-cran-med data set

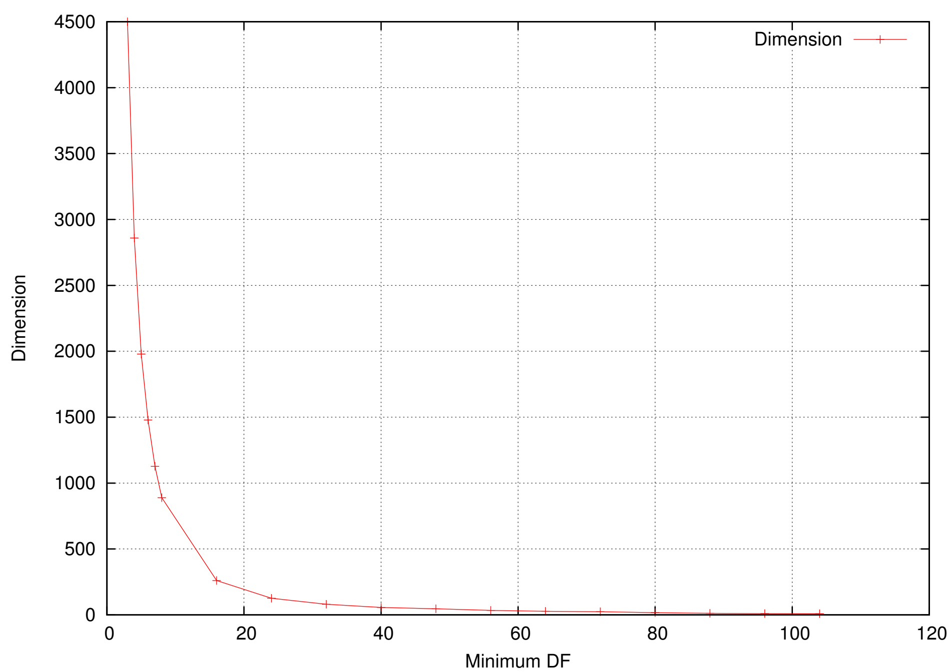

Figure 4.14: Dimensionality of feature space as a function of minimum document frequency for feature selection with term representation on the Reuters data set

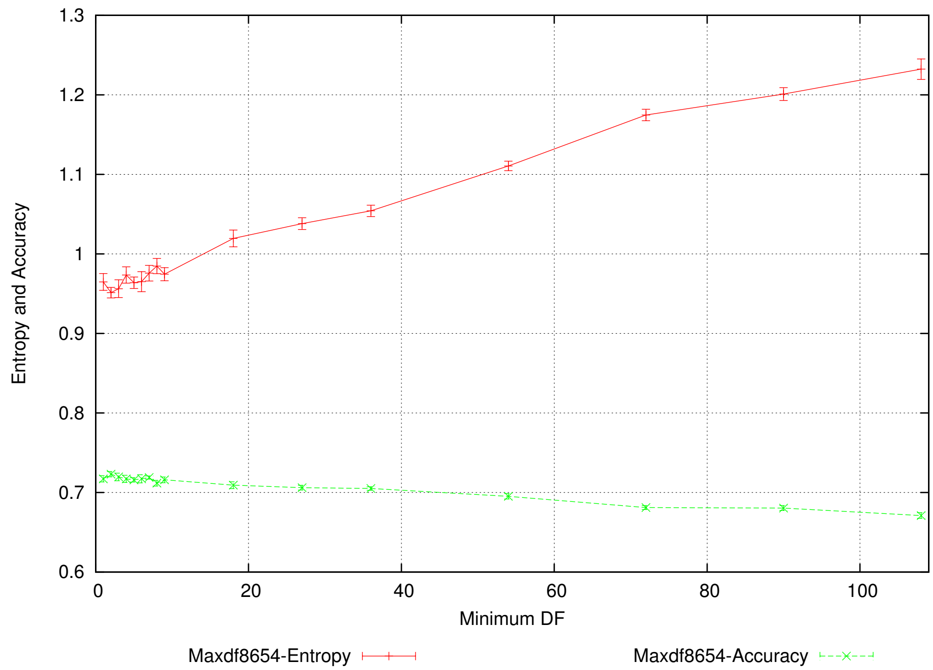

Figure 4.15: Clustering performance using word representation measured by entropy and accuracy as a function of the minimum document frequency for feature selection on the Reuters data set

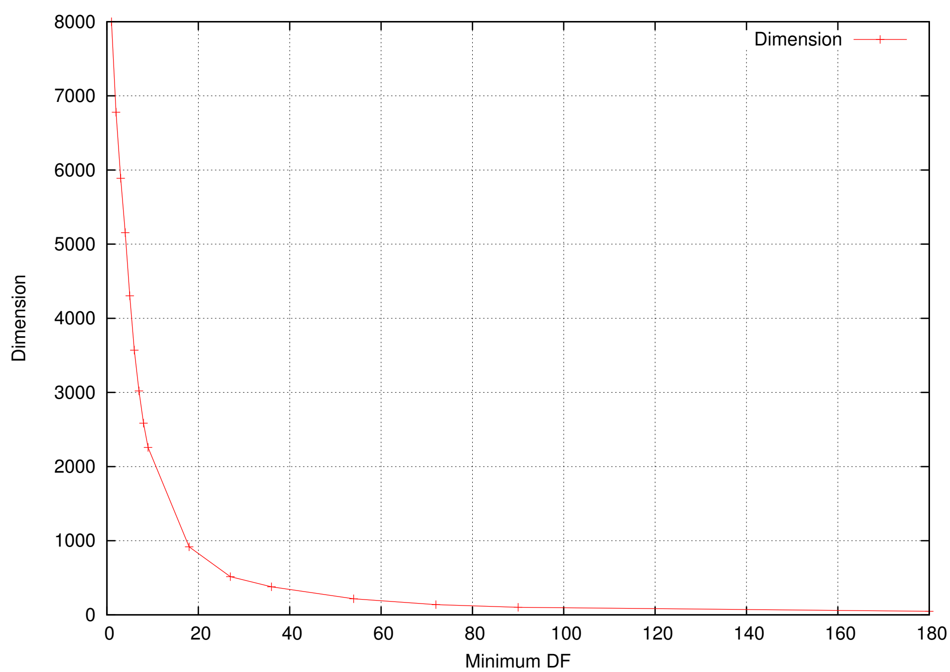

Figure 4.16: Dimensionality of feature space as a function of minimum document frequency for feature selection with word representation on the Reuters data set

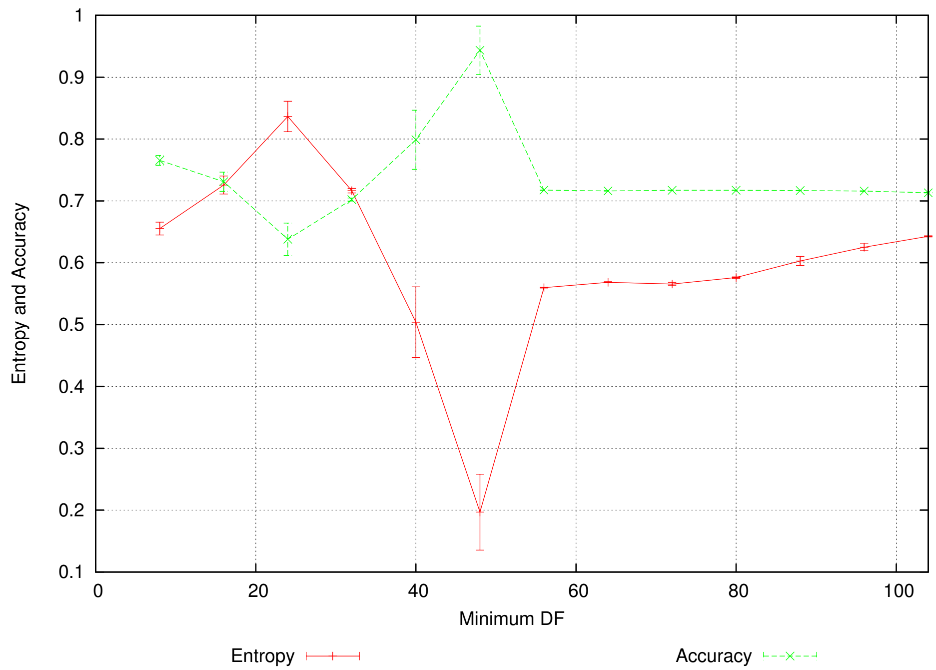

Figure 4.17: Clustering performance using word representation measured by entropy and accuracy as a function of the minimum document frequency for feature selection on the cisi-cran-med data set

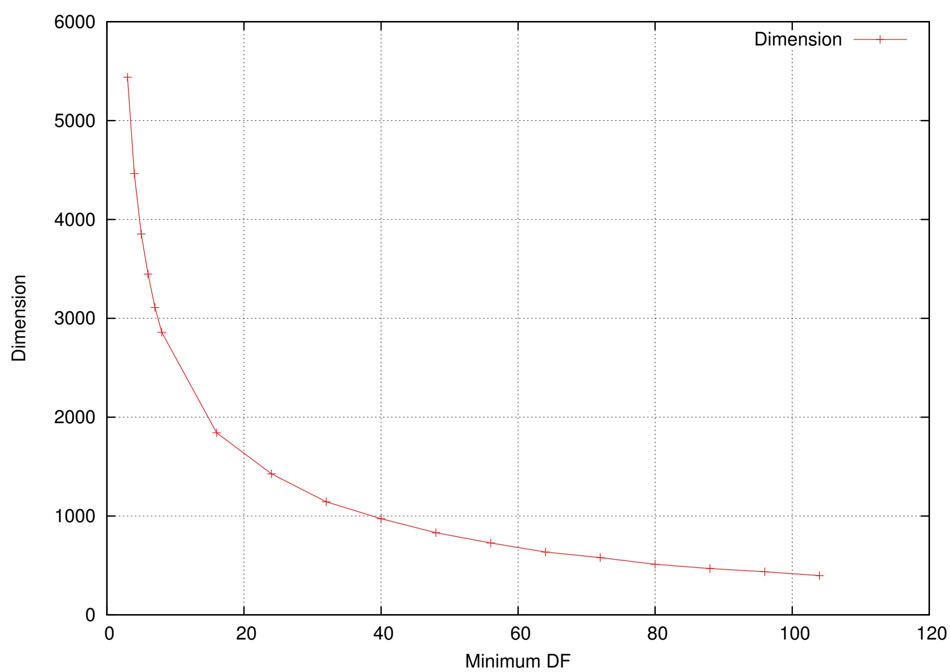

Figure 4.18: Dimensionality of feature space as a function of minimum document frequency for feature selection with word representation on the cisi-cran-med data set

4.3.3 Comparing document clustering using N-grams, terms and words in different dimensionality

We performed experiments on comparing clustering performance using Tri-gram, term and word representation with different dimensionality. To get lower dimensionality for Tri-grams, first we choose the Tri-grams with maximum df = (Total number of documents) / 2; second, we sorted Tri-grams with their minimum df in the reverse order(from the largest to the smallest) and choose the first n Tri-grams. For term-based representation, we use *Filter 1*, *C-Value* and choose the n highest *C-Value* terms. For word-based representation, we use the n most frequent words. The results are the means of 30 times running to reduce the impacts from random seeds.

Tri-gram representation, as shown in Table 4.8, Table 4.9, Table 4.10 and Table 4.11, produces the best clustering results on both Reuters and cisi-cran-med data sets. Even the worst clustering result using Tri-gram representation is better than the best results using term or word representation. Term representation gives better results than word representation on the Reuters data set, while the only exception happens when dimensionality is 8,000. On the cisi-cran-med data set, however, term representation gives worse clustering quality than word representation.

Dimensionality has impacts on the clustering quality. On the Reuters data set, generally, we get better clustering quality if we increase the dimensionality from 1,000 to 8,000. On the cisi-cran-med data set, however, the dimensionality = 2,000 gives the best clustering result.

Dimensionality	Tri-gram	Term	Word
1000	0.763	1.023	1.107
2000	0.726	0.985	1.215
4000	0.805	0.971	1.168
6000	0.761	0.969	1.140
8000	0.689	0.965	0.968

Table 4.8: Comparing clustering performance measured by entropy using Tri-grams, Terms and Words representation w.r.t. different dimensionality on the Reuters data set

One tail T-test is used to prove one result is really better than another result, since from our inspection on some smaller data sets we observed that clustering using N-grams gives better results than using terms and clustering using terms gives better results than using words. That is, the alternative hypothesis is one result is better than the other and null hypothesis is the alternative hypothesis is not true.

Dimensionality	Tri-gram	Term	Word
1000	0.783	0.709	0.697
2000	0.789	0.713	0.657
4000	0.765	0.714	0.673
6000	0.778	0.718	0.682
8000	0.803	0.719	0.717

Table 4.9: Comparing clustering performance measured by accuracy using Tri-grams, Terms and Words representation w.r.t. different dimensionality on the Reuters data set

Dimensionality	Tri-gram	Term	Word
1000	0.455	0.704	0.647
2000	0.186	0.677	0.690
3000	0.443	0.696	0.649
4000	0.484	0.689	0.485

Table 4.10: Comparing clustering performance measured by entropy using Tri-grams, Terms and Words representation w.r.t. different dimensionality on the cisi-cran-med data set

Dimensionality	Tri-gram	Term	Word
1000	0.796	0.668	0.731
2000	0.955	0.675	0.749
3000	0.840	0.663	0.787
4000	0.827	0.659	0.850

Table 4.11: Comparing clustering performance measured by accuracy using Tri-grams, Terms and Words representation w.r.t. different dimensionality on the cisi-cran-med data set

The result one tail T-test for clustering performance using Tri-grams and terms, as shown in Table 4.14 and Table 4.15, confirms that clustering performance using Tri-grams is significantly better than using terms on both of the Reuters and cisi-cran-med data sets, since all p-value are much smaller than 0.05. Moreover, term representation generally gives significantly better results than word representation on the Reuters data set, as shown in Table 4.14. However, there is an exception on the accuracy measure when dimensionality equals 8,000. On the other hand, there are not significant differences between the results of term and word representation when dimensionality is 1,000 or 2,000 on the cisi-cran-med data set, as shown in Table 4.15. When the dimensionality is 3,000 or 4,000, we see that word representation produces better clustering quality than term representation on the cisi-cran-med data set.

Dimensionality	Entropy	Accuracy
1000	0.000	0.000
2000	0.000	0.000
4000	0.000	0.000
6000	0.000	0.000
8000	0.000	0.000

Table 4.12: The p-value of t-test results for clustering performance measured by entropy with Tri-grams and Terms representation w.r.t. different dimensionality on the Reuters data set

Dimensionality	Entropy	Accuracy
1000	0.000	0.000
2000	0.000	0.000
3000	0.000	0.000
4000	0.001	0.001

Table 4.13: The p-value of t-test results for clustering performance measured by entropy with Tri-grams and Terms representation w.r.t. different dimensionality on the cisi-cran-med data set

Another way to reduce dimensionality is random projection. However, the way of using random projection give very worse result especially with the N-gram representation on the cisi-cran-med data set, which are shown in Table 4.16 and Table 4.17. For the N-gram representation, it gives the results nearly as bad as the results of randomly assigning a document to a cluster. The possible reasons of the bad results may be that random projection suppose the distance measure is Euclidean distance. So what we get from random projection has nothing related to the normalized N-gram frequencies and the Eq. 3.2 cannot be used. Also word representation gives

Dimensionality	Entropy	Accuracy
1000	0.000	0.000
2000	0.000	0.000
4000	0.000	0.000
6000	0.000	0.000
8000	0.302	0.378

Table 4.14: The p-value of t-test results for clustering performance measured by entropy with Terms and Words representation w.r.t. different dimensionality on the Reuters data set

Dimensionality	Entropy	Accuracy
1000	0.368	0.013
2000	0.033	0.215
3000	0.041	0.001
4000	0.000	0.000

Table 4.15: The p-value of t-test results for clustering performance measured by entropy with Terms and Words representation w.r.t. different dimensionality on the cisi-cran-med data set

the best results, they are still worse then using the most frequent words to reduce dimensionality. Due to the lack of time, we have not performed random projection with Reuters data set

Dimensionality	Tri-gram	Term	Word
1000	1.087	0.942	0.770
2000	1.087	0.900	0.778
3000	1.087	0.904	0.764
4000	1.087	0.903	0.766

Table 4.16: Clustering performance measured by entropy with character Tri-gram, term and word representation on cisi-cran-med data set. Random project is used to reduce dimensionality.

Dimensionality	Tri-gram	Term	Word
1000	0.377	0.572	0.632
2000	0.377	0.601	0.623
3000	0.376	0.590	0.627
4000	0.377	0.590	0.620

Table 4.17: Clustering performance measured by accuracy with character Tri-gram, term and word representation on cisi-cran-med data set. Random project is used to reduce dimensionality.

Chapter 5

Conclusion and Future work

5.1 Conclusion

In this thesis, we presented a novel approach for document clustering. We use normalized character N-gram frequencies to define documents vectors and a new equation to measure distance between two vectors. The data set used in this theses are Reuters-21578 and cisi-cran-med, while the clustering algorithm is K-Means.

From our experimental results, we see that N-gram representation can be used for document clustering with the new distance measure. Compared with term or word representation, character N-gram representation gives the best clustering results with much lower dimensionality. Term representation gives better results than word representation only on Reuters data set. The experimental results also shows that character N-grams give better clustering results with lower dimensionality than byte N-grams. Moreover, the dimensionality and sparsity of the document matrix increases while N-gram size is increased. The character tri-gram representation compromise the clustering quality and dimensionality well.

In addition, The feature selection based on document frequency can be applied to document clustering with N-gram representation. It can effectively reduce running time by reducing dimensionality without significantly decreasing clustering quality if the maximum and minimum document frequency are chosen right. The random projection, however, produces pool clustering quality with N-gram representation.

Finally, we showed that document clustering using terms, which are chosen automatically based on their *C/NC Value*, gives better clustering result than using words only on Reuters data set.

5.2 Future work

The following topics are of considerable research interest for future work:

- In this thesis, K-Means algorithm is used as the base clustering method. We would like to perform more experiments with other clustering methods. Since K-means algorithm is a partition clustering algorithm for hard clustering, we would like to test heuristic and soft clustering method.

- We used Reuters and cisi-cran-med data sets. N-gram representation produces the best clustering results on both of them. Term representation, however, gives different results compared with word representation. More data sets can be used to compare these three representation.

- To reduce the dimensionality, we used feature selection based on document frequency and random projection. We are interested in applying other existing feature selection methods on N-gram representation.

- We tested clustering results with different dimensionality and the lowest dimensionality is 1,000. We do not know what will happen if we continue reducing dimensionality to only, say, 100. It is possible that a better result will happen with a lower dimensionality.

- We implemented K-Means on C++, which is much faster then Weka, where Java is used as programing language. However, the current implementation does not utilize the properly of the sparse document matrix. We believe that current implementation of K-Means can be speed up and use less memory if a data structure is used, which is efficient with the sparse document vector or matrix.

Bibliography

[1] Cisc-cran-med data set. Accessed in August 2004. http://www.cs.dal.ca/ eem/6505/lab/dataText/.

[2] Florian Beil, Martin Ester, and Xiaowei Xu. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442. ACM Press, 2002.

[3] Paul S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99. Morgan Kaufmann Publishers Inc., 1998.

[4] E. Brill. A simple rule-based part of speech tagger. In *3rd conference on applied natural language processing(ANLP-92)*, pages 152–155, 1992.

[5] William B. Cavnar. Using an n-gram-based document representation with a vector processing retrieval model. In *TREC-3*, pages 269–278, 1994.

[6] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *The 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, August 1996.

[7] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. Automatic recognition of multi-word terms: the C-value/NC-value method. *International Journal on Digital Libraries*, 3(2):115–130, August 2000.

[8] Greg Hamerly and Charles Elkan. Learning the k in k-means. In *Proceedings of the seventeenth annual conference on neural information processing systems*, December 2003.

[9] E. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering in a high-dimensional space using hypergraph models. Technical report, Department of Computer Science and Engineering/Army HPC Research Center, University of Minnesota, 1997.

[10] Jiawei Han and Micheline Kamber. *Data Mining - Concepts and Techniques.* Morgan Kaufmann Publishers, San Mateo, California, 2001.

[11] Eibe Frank Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann Publishers, 1999.

[12] George Karypis. Cluto - clustering toolkit, 2003. Accessed in May 2004. http://www-users.cs.umn.edu/ karypis/cluto/download.html/.

[13] Vlado Kešelj. Perl package Text::Ngrams, 2003. Accessed in May 2004. `http://www.cs.dal.ca/~vlado/srcperl/Ngrams` or `http://search.cpan.org/author/VLADO/Text-Ngrams-1.1/`.

[14] Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. In *PACLING'03*, pages 255–264, August 2003.

[15] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22. ACM Press, 1999.

[16] Anton Leuski. Evaluating document clustering for interactive information retrieval. In *CIKM*, pages 33–40, 2001.

[17] David D. Lewis. Reuters-21578 text categorization test collection distribution 1.0, 1999. Accessed in Octorber 2003. http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[18] J MacQueen. Some methods for classification and analysis of multivariate observation. In *The fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[19] MATLAB. kmeans.m. Accessed in April 2004. /opt/matlab-R13/toolbox/stats at flame.cs.dal.ca.

[20] E. Milios, Y. Zhang, B. He, and L. Dong. Automatic term extraction and document similarity in special text corpora. In *PACLING'03*, pages 275–284, August 2003.

[21] Beckmann N., Kriegel H.-P., and Seeger B. R*-tree: An efficent and robust access method for points and rectangles. In *ACM SIGMOD intenational conference on management of data*, pages 322–331, 1990.

[22] M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997.

[23] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[24] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[25] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[26] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000), 30-31 July 2000, Austin, Texas, USA*, pages 58–64. AAAI, July 2000.

[27] Zhong Su, Qiang Yang, HongHiang Zhang, Xiaowei Xu, and Yuhen Hu. Correlation-based document clustering using web logs. In *HICSS*, 2001.

[28] E. Milios Y. Zhang, N. Zincir-Heywood. Term-based clustering and summarization of web page collections. In *the Seventeenth Conference of the Canadian Society for Computational Studies of Intelligence (AI'04)*, pages 60–74, London, ON, May 2004.

[29] Lingyan Zhang. Parallel automatic term extraction from large web corpora. Master's thesis, Faculty of Computer Science, Dalhousie University, Halifax, NS, 2004.

[30] Zipf and George K. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley Press Inc., Cambridge 42, MA, 1949.