Scalability issue in mining large data sets

A. Mc Manus & M.-T. Kechadi
Department of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland

Abstract

The most distinct characteristic of data mining is that it deals with large data sets. This requires the algorithms used in data mining to be highly scalable. However, most algorithms currently used in data mining do not scale very well when applied to very large data sets because they were initially developed and tested upon smaller data sets. Today, we have such large data sets that these algorithms are no longer efficient enough for mining and analysing. In this paper, we have addressed a data clustering problem and developed a scalable algorithm, which provides very high precision and recall values. This algorithm is used on a very large real-world data-set, TDT2, and experimental results showed that this algorithm performs well compared to traditional ones.

1 Introduction

Nowadays, people have great demand on knowledge and information, while information overload becoming one serious problem. News media and publishing industries therefore try to suit customers needs by using electronic information management system. Document clustering algorithm has been introduced to group similar documents together for easier searching and reading. Document clustering algorithm has been widely used in news media and publishing industry,which ensured it effectiveness over manual clustering. With labor cost reduced and time saved, document clustering algorithms provides convenient clustered-news for users.

Clustering is an important task that is performed as part of many text mining and information retrieval systems. Clustering can be used for efficiently finding the nearest neighbors of a document [1, 4], for improving the precision or recall in information retrieval systems, for aid in browsing a collection of documents [2],

Data Mining V, A. Zanasi, N. F. F. Ebecken & C. A. Brebbia (Editors)

for the organization of search engine results [2, 6], or for the personalization of search engine results.

Recently there has been much interest in the problem of similarity search in time series databases, which is our main insert in this paper. This is hardly surprising given that time series account for much of the data stored in business, medical and scientific databases. Similarity search is useful in its own right as a tool for exploring time series databases, and it is also an important subroutine in many KDD applications such as clustering [2, 4, 6], classification [5] and mining of association rules.

Different document clustering methods have been examined. These conventional clustering methods mainly consist of two parts: construction of a similarity matrix between documents and formulation of clustering algorithm to generate clusters.

2 Mining large datasets

In the present times we are witnessing an explosive growth in the amount of data that is being collected in the business and scientific arena. Data warehouses are filling up with huge amounts of data in every conceivable form. In most cases, the sheer size of the datasets prevents the vast majority of the data from being deeply analyzed. Often large portions may not have been examined at all. The field of Data Mining (or Knowledge Discovery in Databases) attempts to develop automatic procedures that search these enormous data sets to obtain useful information that would otherwise remain undiscovered. Such new knowledge can take the form of patterns, rules, clusters, or anomalies that exist in the massive datasets. These discoveries could potentially be of great significance to scientific or business organizations. Given the size and dimensionality of the datasets, high performance algorithms and systems are an integral component of a successful data mining solution.

2.1 The corpus

Throughout all experimental results given in this paper, the TDT pilot corpus was used. There is about 16,000 stories, with half taken from Reuters newswire and the other from CNN broadcast news transcripts. The transcripts are arranged in chronological order, and structured in SGML format, and are available from the Linguistic Data Consortium (LDC).

2.2 Heterogeneity of the data

Many steps are taken in the data pre-processing stage: heterogeneity resolution, data cleansing, data transformation, and data reduction before any algorithm can be introduced to produce quality clusters. In order to achieve data which we can work upon, we apply some mining tools, which help in the extraction of patterns from the pre-processed data.

Data Mining V, A. Zanasi, N. F. F. Ebecken & C. A. Brebbia (Editors)

2.3 Preprocessing of data

One major drawback when mining such large amounts of data is that documents contain a lot of words most of which are repeated. Using all the words in a collection of documents to represent those documents would certainly result in high inefficiency that could be eliminated with some pre-processing steps.

Before any clustering is performed, a sequence of preprocessing steps were taken in order to optimize performance. The reason for those steps is to reduce the number of words for mining. A list of those tasks follows next:(1)retrieve data - from a variety of heterogeneous operational databases (TDT Pilot in our case), (2) data is transformed and delivered to the data warehouse/store based on a selected model (or mapping definition), (3) metadata - information describing the model and definition of the source data elements, (4) data cleansing - removal of certain aspects of operational data, such as low-level transaction information, which slow down the query times, (5) transfer - processed data transferred to the data warehouse, a large database on a high performance box

2.3.1 Stop-word removal

Stop words are words that have no significant semantic content in the context in which it exists. Stop words can also be words that have high frequencies across a set of collection of document. A list containing all the stop words in a system or application is referred to as a stop list. Stop words are not included as indexing terms. For example, the words “the”, “on”, “with” are usually stop words.

2.3.2 Stemming

Stemming means reducing different word forms to common roots. The purpose of stemming is to group words that are morphological variants on the same word stem. For example, the words “Computing”, “Computer”, and “Computational” all map to “Compute”. This has the advantages of eliminating suffixes indicating tag-of-speech, verbs and/or plural forms; however, stemming algorithms employ a great deal of linguistics and are language-dependent. This technique is also known as suffix stripping or term truncation (van Rijsbergen 1979; Salton 1989). In the English language, stemming is traditionally carried out by stripping off the common suffixes, and the Porter and the Lovins stemmer are two widely used implementations. The Lovins Stemmer, implemented within this system, removes a maximum of one suffix from a word, due to its nature as single pass algorithm. It uses a list of about 250 different suffixes, and removes the longest suffix attached to the word, ensuring that the stem after the suffix has been removed is always at least 3 characters long. Then the ending of the stem may be reformed (e.g., by un-doubling a final consonant if applicable), by referring to a list of recoding transformations.

The value of stemming has been questioned in the past, but recent studies have shown that stemming can produce consistent though small improvements in retrieval effectiveness over a large range of collections. When dealing with relatively short documents, stemming has even higher value. Since many of the

Data Mining V, A. Zanasi, N. F. F. Ebecken & C. A. Brebbia (Editors)

documents we are dealing with are very short, this indicates that the use of a stemming component could prove valuable.

3 Cluster analysis

Cluster Analysis, also called data segmentation, has a variety of goals. All relate to grouping or segmenting a collection of objects (also called observations, individuals, cases, or data rows) into subsets or "clusters", such that those within each cluster are more closely related to one another than objects assigned to different clusters. Central to all of the goals of cluster analysis is the notion of degree of similarity (or dissimilarity) between the individual objects being clustered. There are two major methods of clustering – hierarchical clustering and k-means clustering. *So why cluster again?*. Cluster analysis divides data into groups (clusters) for the purposes of summarization or improved understanding also. For example, cluster analysis has been used to group related documents for browsing, to find genes and proteins that have similar functionality, or as a means of data compression.

In hierarchical clustering the data are not partitioned into a particular cluster in a single step. Instead, a series of partitions takes place, which may run from a single cluster containing all objects to n clusters each containing a single object. Hierarchical Clustering is subdivided into agglomerative methods, which proceed by series of fusions of the n objects into groups, and divisive methods, which separate n objects successively into finer groupings.

In this paper we present two of the different data mining techniques that were used in the system. Both algorithms clustered various sized datasets and the scalability, efficiency and accuracy of each was compared to one another. The algorithms used were Single-link and two different hierarchical clustering methods, utilising different variations of the cosine similarity function, mainly focusing on the latter because agglomerative techniques are more commonly used.

3.1 Agglomerative hierarchical clustering

An agglomerative hierarchical clustering procedure produces a series of partitions of the data, Pn, Pn-1, ... , P1. The first Pn consists of n single object 'clusters', the last P1, consists of single group containing all n cases.

At each particular stage the method joins together the two clusters which are closest together (most similar). (At the first stage, of course, this amounts to joining together the two objects that are closest together, since at the initial stage each cluster has one object.)

Differences between methods arise because of the different ways of defining distance (or similarity) between clusters. Several different types of cosine similarity will be compared in detail in this paper.

Data Mining V, A. Zanasi, N. F. F. Ebecken & C. A. Brebbia (Editors)

3.2 Single-pass clustering

One of the simplest agglomerative hierarchical clustering method is single linkage, also known as the nearest neighbor technique. The defining feature of the method is that distance between groups is defined as the distance between the closest pair of objects, where only pairs consisting of one object from each group are considered. In the single linkage method, D(r,s) is computed as D(r,s) = Min d(i,j) : Where object i is in cluster r and object j is cluster s .

Here the distance between every possible object pair (i,j) is computed, where object i is in cluster r and object j is in cluster s. The minimum value of these distances is said to be the distance between clusters r and s. In other words, the distance between two clusters is given by the value of the shortest link between the clusters. At each stage of hierarchical clustering, the clusters r and s , for which D(r,s) is minimum, are merged.

The approach is fairly fast and result in hierarchies where the closest nearest neighbors are at lower levels of the hierarchy. However,it leads to non-balanced clusters,and many node-node comparisons can have the same strength of similarity thus many documents can be linked at the same level in the hierarchy.

3.3 Similarity analysis

One important issue in our approach is the problem of determining the right similarity function. The critical property of the similarity function is its ability to separate stories that discuss the same topic from a given query documents from those that have different related topics.

We must first compute the tf*idf of each test dataset. This score is defined as follows: For each term i in a document j, we compute the term frequency of i in j, i.e. the number of occurrences of i in j (tf i,j). We then compute the inverse document frequency of term i by taking the log of the ratio of N, the number of documents in our corpus, to the document frequency of i, i.e. the number of documents in which term i occurs (dfi). A term which occurs in all documents will have inverse document frequency 0. A term which occurs very often in one document but in very few documents of the corpus will have a high inverse document frequency and thus a high tf*idf score, and is thus a strong candidate for being a term that characterizes the content of the document.

We next represent each document by a vector made up of the tf*idf score wi for each term i. By comparing these vectors, we can arrive at a quantitative measure of the similarity of two texts to one another. The most interesting metric for these purposes seems to be the cosine similarity measure, defined as the dot product of the two vectors divided by the product of the length of the vectors. The cosine similarity measure of two texts varies from 0 (no terms in common) to 1 (every term in both texts has the same tf*idf score).

We will look at three different functions of the Cosine Similarity function in this paper: Euclidean, Raw Cosine and my own function.

Data Mining V, A. Zanasi, N. F. F. Ebecken & C. A. Brebbia (Editors)

3.3.1 Cosine similarity

Cosine similarity is a classic measure used in IR, and is consistent with a vector-space representation of stories. the measure is simply an inner product of two vectors, where each vector is normalized to unit length.

Cosine seems to perform best at full dimensionality, as in the case of comparison of two long stories. Performance degrades as one of the vectors becomes shorter. Because if the normalization, cosine is less dependent on specific term weights, and performs well when raw word counts are presented as weights.

Definition 1 Euclidean similarity

Given the vector space defined by all terms in the document pool and the query, compute the Euclidean distance z between each document d and the query q. The similarity between d and q, if M is the maximum distance for all documents in the pool, is 1 - (z / M). Euclidean similarity is always computed with Normalization = false and TFMode = 0.

Definition 2 Raw cosine similarity

Given that df(i) is the frequency of term i in a document, and qf(i) is the frequency of term i in a query, raw cosine similarity is defined over all terms occurring in both the document and the query as:

We must first compute the tf*idf(F)of each test dataset. This score is defined as follows: For each term i in a document j, we compute the term frequency of i in j, i.e. tf(j,i). We then compute the inverse document frequency of term i by taking the log of the ratio of N, the number of documents in our corpus, to the document frequency of i, i.e. the number of documents in which term i occurs df(i).

$$RawCosineSim = \frac{\sum_{1=n}^{n} df(i) * qf(i)}{(\sqrt{(\sum_{1=n}^{n} df(i)^2)} * (\sqrt{(\sum_{1=n}^{n} qf(i)^2)})} \tag{1}$$

However, our version of the "Raw Cosine similarity" makes a difference that achieves a higher accuracy in document retrieval. By using the tf*idf scores for elements of **each vector**, it computes the entire dot product of the two vectors and not just the dot product of the similar sections as in "Raw Cosine similarity". So the dot product is:

$$SC\text{-}Cosine = \frac{\sum_{1=n}^{n} F(i)^2 (df(i) * qf(i))}{(\sum_{1=n}^{n} F(i) * df(i)^2))^{1/2} * ((\sum_{1=n}^{n} F(i) * qf(i)^2))^{1/2}} \tag{2}$$

We propose an explanation for these results that is based on an analysis of the clustering algorithm specified by equation(2) and the nature of document data.

Data Mining V, A. Zanasi, N. F. F. Ebecken & C. A. Brebbia (Editors)

3.4 Detection of threshold

Despite the fact that keywords are not always good descriptors of contents, most existing search engines still rely solely on the keywords contained in documents and queries to calculate their similarity. This is one of the key factors that affect the precision of the search engines. In many cases, the answers returned by search engines are not relevant to the user's information need, although they do contain the same keywords as the query. To study the effect of different Cosine functions in conjunction with the Single-link algorithm and our algorithm, that of Scatter/Gather [2] incorporated with SC-Cosine ,we calculated the percentage of recall and precision under different threshold values. To detect this optimal threshold, we evaluated a set of experimental datasets, varying from 50 up to 1600 documents. It was found that 0.21 was the threshold that we received the greatest accuracy when calculating the precision and recall for these clusters.

- Scatter/Gather + SC-Cosine (dataset size = 1600 documents)

Threshold	0.15.	0.17	**0.21**	0.21	0.22	0.23	0.26	0.3	0.31
$Precision$	0.67	0.71	**0.93**	0.89	0.88	0.85	0.79	0.83	0.74
$Recall$	0.31	0.42	**0.48**	0.47	0.46	0.46	0.44	0.43	0.42

- Single-Link with Raw Cosine Similarity (dataset size = 1600 documents)

Threshold	0.12	0.15	**0.16**	0.17	0.19	0.21	0.23	0.25	0.27
$Precision$	0.48	0.49	**0.53**	0.52	0.47	0.45	0.45	0.44	0.40
$Recall$	0.58	0.62	**0.78**	0.75	0.59	0.58	0.56	0.55	0.51

4 Results analysis

Many experiments were carried out with document sets of different sizes which were taken from the above-mentioned corpus.Precision and recall have been and continue to be very useful measures of performance for information retrieval and extraction. Precision deals with substitution and insertion errors while recall deals with substitution and deletion errors. Performance metrics are based on Recall and Precision,which are defined as follows:

	In Event	Not in Event
$InCluster$	A	B
$NotinCluster$	C	D

$$Recall = A/(A + C) if A + C > 0 \tag{3}$$

$$Precision = A/(A + B) if A + B > 0 \tag{4}$$

Data Mining V, A. Zanasi, N. F. F. Ebecken & C. A. Brebbia (Editors)

Because of our desire to have a single measure of performance that deals with all three types of errors simultaneously, the F-measure was defined [3]. The F-measure is defined as:

$$F\text{-}measure = 2(PR)/P + R \tag{5}$$

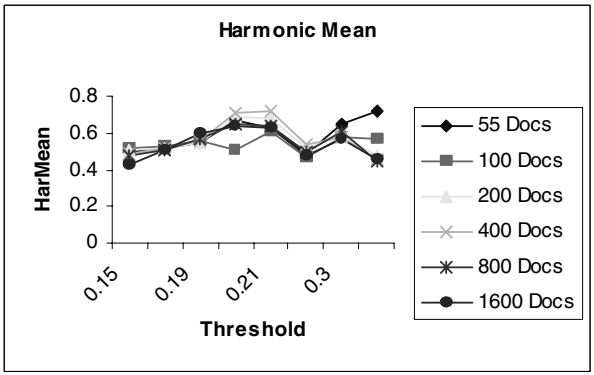

Figure 1: Harmonic Mean.

In Figure 1 the graph shows that both recall and precision values decrease with increasing threshold. The performance of such algorithms as Single-Link and our algorithm, Scatter/Gather [2] with SC-Cosine implemented, is better when the threshold is smaller. When tested on the 6th dataset (1600 documents) the precision values obtained rise as the threshold increases until it reaches a peak at the threshold level of 0.21. The precision obtained at this peak is 0.93, which is higher that previous experiments, averaging about 0.8/0.84 [4]. Recall exhibits a similar trend and peaks at the same threshold value of 0.21, a recall value of 0.48. The recall value calculated for threshold values from 0.2 up to 0.4 are seen to slowly drop, but after 0.4 there is a dramatic decrease in the values. From Figure 1, we can see that the Harmonic Mean is at a peak at 0.21. Above this threshold we notice a slow decrease, because as we increase the data size, there is an increased chance of data been not similar and been wrongly added to existing clusters, resulting in a lower precision/recall. This leads to the conclusion that it is at this threshold of 0.21 that the most accurate results are achieved.

5 Conclusion

After several experiment were carried out, using up to 1600 news articles from the corpus, we notice that when using such large amounts of data, the threshold can drop dramatically by changing the threshold throughout. Our results are relatively close to the results of TDT Pilot study, however, the accuracy of our approach, in

relation to Precision and Recall at the optimum threshold, is a significant improvement.

Acknowledgements

We wish to express our appreciation to those whose publications have assisted us with our research. We would also like to thank the referees for their comments which helped the improvement of this paper.

References

[1] C. Buckley and A. Lewit. Optimizations of inverted vector searches. In SIGIR, pages 97–110, 1985.
[2] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 318–329, 1992.
[3] A. Debregeas and G. Hebrail. Interactive interpretation of kohonen maps applied to curves. In 4th International Conference of Knowledge Discovery and Data Mining, pages 179–183, 1998.
[4] S. Kennedy and T. Kechadi. Examining the scability of k-means in high dimensional datasets. 2003.
[5] E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In 4th International Conference of Knowledge Discovery and Data Mining, pages 239–241, 1998.
[6] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and intuitive clustering of web documents. In Knowledge Discovery and Data Mining, pages 287–290, 1997.