TDT- An Efficient Clustering Algorithm for Large Database

Ms. Kritika Maheshwari, Mr. M.Rajsekaran

M-Tech Scholar, Department of Computer Science and Engineering, SRM University, India

Assistant Professor, Department of Computer Science and Engineering, SRM University, India

ABSTRACT: *A lot of side-information is available along with the text documents in online forums. Such side information may be of different kinds, as it may be the links in the document, access behavior from web histories or other non-textual attributes which are embedded into the text document. Such attributes contain huge amount of information for clustering purposes. However, the importance of this side-information is difficult to calculate, mostly when some of the information is noisy. Therefore in these cases it is risky to incorporate side information into the clustering process, because it may either improve the quality of the clustering process, or it can even add some noisy information to it. Therefore, a principled way to perform the clustering process is needed, so as to maximize the advantages from using this side information. And to result the search query efficiently and effectively. An algorithm for text clustering with side-information is described here i.e. COATES Algorithm.*

Keywords*-Clustering process, Dimensionality reduction, Side Information, Topic Detection tracking.*

1. INTRODUCTION

In recent years, new ways of collecting data have resulted in a need for applications which work effectively and efficiently with data streams. The most important problem in the data stream domain is that of clustering. We will examine the problem of massive domain stream clustering. Massive-domains are those data domains in which the number of possible values for one or more attributes is very large. Examples of such domains are, in network applications, many attributes like IP addresses are drawn over millions of possibilities. In a multi-dimensional application, this problem is further magnified because of the multiplication of possibilities over different attributes.

In text clustering, a text or document is always represented as a bag of words. This representation involves one major problem: the high dimensionality of the feature space and the inherent data sparsity. Obviously, a single document has a sparse vector over the set of all terms. The performance of such clustering algorithms will decline dramatically due to the problems of high dimensionality and data sparseness. Therefore it is highly desirable to reduce the feature space dimensionality. There are two commonly used techniques to deal with this problem: feature extraction and feature selection.

In many real data mining applications, data comes in as a continuous stream and presents several challenges to traditional static data mining algorithms. Application examples include topic detection from a news stream, intrusion detection from continuous network traffic, object recognition from video sequences, etc. Challenges lie in aspects such as: high algorithm efficiency is required in real time; huge data volume that cannot be kept in memory all at once; multiple scans from secondary storage is not desirable since it causes intolerable delays; and mining algorithms need to be adaptive since data patterns change over time.

In many application domains, a tremendous amount of side-information is also associated along with the documents. Since text documents mostly occur in the context of a variety of applications in which there may be a large amount of other kinds of database attributes or meta information which may be useful to the clustering method.

2. RELATED WORK

In the paper [1] Charu C Agrawal discuss about massive-domain data streams are those in which the number of possible domain values for each attribute are very large and cannot be easily tracked for clustering purposes. Examples of such streams include IP-address streams, streams of credit-card transaction or streams of sales data over large numbers of items. So in these cases, it is well known that simple stream operations like counting 000can be extremely difficult because of the difficulty in maintaining summary information over the different distinct values. The clustering task is significantly more challenging in such cases, as the intermediate statistics for the different clusters is difficult to be maintained efficiently.

Here, Charu C. Aggarwal proposes a method for clustering massive-domain data streams with the use of sketches. Charu C. Aggarwal proves probabilistic results which show that a sketch-based clustering method can provide similar results to an infinite space clustering algorithm with high probability. The problem of massive-domain clustering naturally occurs in the space of discrete attributes, whereas most of the known data stream clustering methods is designed on the space of continuous attributes. He proposes a sketch-based approach in order to keep track of the intermediate statistics of the primary clusters. These statistics are used to make approximate determinations of the assignment of data points to clusters.

He also provides probabilistic results[2] which indicate that these approximations are sufficiently accurate to provide similar results to an infinite-space clustering algorithm with high probability. He also presents experimental results which illustrate the high accuracy of the approximate assignments. In the next section, we will propose a technique for massive-domain clustering of data streams. He provides a probabilistic analysis which shows that our sketch-based stream clustering method provides similar results to an infinite space clustering algorithm with high probability. And this literature from the title of A Framework for Clustering Massive-Domain Data Streams.

In many real data mining applications, data comes in as a continuous stream and presents several challenges to traditional static data mining algorithms. Application examples include topic detection from a news stream, intrusion detection from continuous network traffic, object recognition from video sequences, etc. And challenges lie in several aspects like high algorithm efficiency is required in real time; huge data volume that cannot be kept in memory all at once; multiple scans from secondary storage is not desirable since it causes intolerable delays; and mining algorithms need to be adaptive since data patterns change over time.

In this paper[3] Douglass R Cutting and David R. Kargel discuss that document clustering has not been well received as an information retrieval tool. Problem in its use occur into two main categories , the one where that clustering is too slow for large corpora; and the other, that clustering doesn't appreciably improve retrieval. These problems arise only when clustering is used in an attempt to improve conventional search techniques. Well, considering clustering as an information access tool in its own right prevent these problems, and provides a efficient new access paradigm.

A technique for browsing documents that employs document clustering as its primary operation, also present fast clustering algorithms which support this interactive browsing paradigm.

In the basic iteration of the proposed browsing method, the user is presented with short summaries of a small number of document groups. Initially the system scatters the collection into a small number of document groups, or clusters, and presents short summaries of them to the user. Based on these summaries, the user selects one or more of the groups for further study. The selected groups are combined together to form a sub collection.

The system then applies clustering again to scatter the new sub collection into a small number of document groups, which are again presented to the user. With each following iteration the groups become smaller, and more detailed. Ultimately, when the groups become smaller, this process bottoms out by enumerating individual documents.

To support Scatter/Gather, fast clustering algorithms are essential. Clustering can be done faster by working in a same manner on small groups of documents rather than trying to deal with the entire corpus globally. For extremely large corpora, even the linear time clustering achieved by the Buckshot or Fractionation algorithms may be too slow. Authors are working to develop variations on Scatter/Gather which will scale to arbitrarily large corpora, under the assumption that linear time pre-processing will always be feasible.

In this paper BIRCH: An Efficient Data Clustering Method for Very Large Database[4] Tian Zhang, Raghu Krishnan and Miron Livny are concerned about finding useful patterns in large datasets has attracted considerable interest recently, and one of the most deeply studied problems in this area is the recognition of clusters, or densely populated areas, in a multi-dimensional Dataset. Prior work does not reasonably address the problem of large datasets and minimization of I/0 costs. Authors adopt, the problem definition used in Statistics, but with an additional, database-oriented constraint, the amount of memory available is limited and we want to minimize the time required for I/0.

A related point is that it is desirable to be able to take into account the amount of time that a user is willing to wait for the results of the clustering algorithm. Authors a clustering method named BIRCH and demonstrates that it is especially suitable for very large databases. Its I/O cost is linear in the size of the dataset: a single scan of the dataset yields a good clustering, and one or more additional passes can (optionally) be used to improve the quality further.

By evaluating BIRCH'S time/space efficiency data input order sensitivity, and clustering quality, and comparing with other existing algorithms through experiments, we argue that BIRCH is the best available clustering method for very large databases. BIRCH architecture also gives opportunities for parallelism, and for interactive or effective performance tuning based on knowledge about the dataset, gained over the course of execution. Finally, BIRCH is the first clustering algorithm proposed in the data areas that address other and propose a plausible solution.

Proper parameter setting is important to BIRCH efficiency. The main things which needed to be considered are more reasonable ways of increasing the threshold dynamically, The dynamic adjustment of outlier criteria, More accurate quality measurements and Data parameters that are good indicators of how well BIRCH is likely to perform.

Authors try to explore BIRCH'S architecture for opportunities of parallel executions as well as interactive learning's. As an incremental algorithm, BIRCH will be able to read data directly from a tape drive or from network by matching its clustering speed with the data reading speed. They also study how to make use of the clustering information obtained to help solve problems such as storage or optimization, and data compression.

In this paper [5] author Shi Zhong combines an efficient online spherical k-means (OSKM) algorithm with an existing scalable clustering strategy to achieve fast and adaptive clustering of text streams. The OSKM algorithm modifies the spherical k-means (SPKM) algorithm, using online update (for cluster centroids) based on the well-known Winner-Take-All competitive learning. It has been proved to be as efficient as SPKM, but it is much superior in the quality of clustering.

3. Proposed System

The scalable clustering strategy was previously developed to deal with very large data bases that cannot fit into a limited memory and that are too expensive to read/scan multiple times. Using this clustering strategy, one keeps sufficient statistics for history data to retain (part of) the contribution of history data and to accommodate the limited memory.

To make the proposed clustering algorithm adaptive to data streams, the author introduce a forgetting factor that applies exponential decay to the importance of history data. As the older a set of text documents, the less weight they carry. This experimental result demonstrate the efficiency of the proposed algorithm and reveal an intuitive and an interesting fact for clustering text streams—one needs to forget to be adaptive.

The primary goal in this paper is to study the clustering problem. This approach can also be extended in principle to other data mining problems in which auxiliary information is available with text. Such side information may be of different kinds, such as the links present in document, user-access behavior from web history, or other non-textual attributes which are embedded into the text document. These attributes contain a enormous amount of information for clustering purposes. The relative importance of this auxiliary information may be difficult to estimate. It can be risky to incorporate side-information into the mining process.

The algorithm for text clustering with side-information is COATES. We assume that an input to the algorithm is the number of clusters k. In all text-clustering algorithms, the stop-words have been removed, and stemming has been performed to improve the unbiased power of the attributes. The algorithm has two phases:

1. Initialization 2. Main Phase.

The TDT (Topic Detection Tracking) is influenced by the prior work of detection. It is found that the dual-threshold clustering paradigm of detection systems worked well for topic detection and tracking . Furthermore, we found that many of the features that were beneficial for topic detection continued to be helpful for tracking. Our basic scoring formula was a symmetrised version of the PCA (Principle Component Analysis) formula. Document clusters were represented by the centroid of the cluster, and the weight is both dependent on cluster and time. The important source of improvement occurs when we include the named based entity search and cluster based representation.

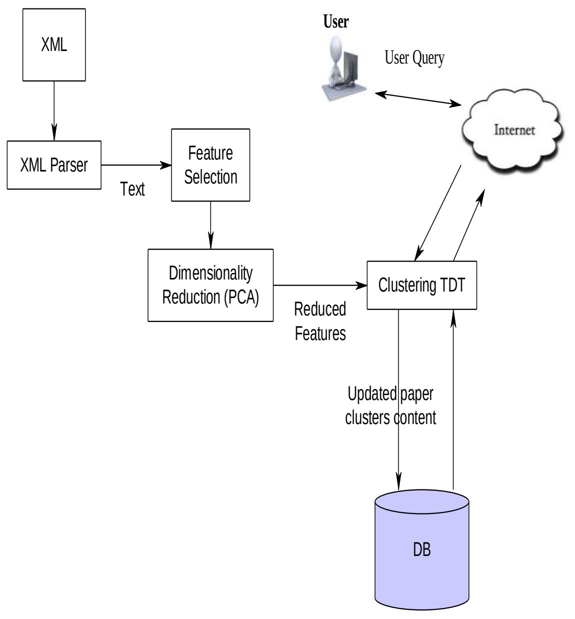

figure.1 System Architecture

In FIG.1 the system architecture is shown where the process is shown how the clustering process is been handled efficiently. The

XML query is been taken from the user by the XML parser. The dimensionality reduction is done using PCA (principal component analysis) which reduce the original content into principal component. The reduced features go to clustering process. The simultaneously the user query is answered.

The main methodology we used is COATES algorithm for clustering with side information and along with it topic based detection is also applied.

We use the COntent and Auxiliary attribute based Text cluStering (COATES) algorithm in text clustering. This algorithm has two phases. Now, this text mining in the initialization phase of the algorithm. We use a lightweight initialization phase in which a standard text clustering approach is used in weblog without any side-information. The centroids and the partitioning created by the clusters formed in the first phase provide an initial starting point for the second phase. As the first phase is based on text only, and it does not have auxiliary information. The focus of the first phase is simply to construct an initialization, which gives a good initialization for the clustering process based on text content.

The main phase of the algorithm is executed after the first phase. The main phase is the combining auxiliary attributes in the text clustering. This phase starts off with these initial groups, and iteratively reconstructs these clusters with the use of both the text content and the auxiliary information. It performs alternating iterations which use the text content and auxiliary attribute information in order to improve the quality of the clustering. And these iterations are known as content iterations and auxiliary iterations respectively. The above two iterations are combined and called as major iteration. The major iteration is the sum of two minor iterations corresponds to text based and auxiliary attribute based.

Here, the overall approach uses alternating minor iterations of content-based and auxiliary attribute-based clustering. The algorithm maintains a set of centroids, which are subsequently refined in the different iterations. In content-based phase, we assign a document to its closest centroid based on a similarity function of text. In auxiliary phase, the attribute probabilities are related to the cluster-membership probabilities by creating the probabilistic model. It relates on the basis of clusters which have already been created in the most recent text-based phase.

The goal of this modeling is to examine the coherence of the text clustering with the side-information attributes. In order to construct a probabilistic model of membership of the data points to clusters, we know that there is a prior probability of each auxiliary iteration for assignment of documents to clusters , and a posterior probability of assignment of documents to clusters with the use of auxiliary variables in that iteration. Furthermore, in order to ensure the robustness of the approach, we need to remove the noisy attributes. When the number of auxiliary attributes is little large, it becomes really important.

We will compute the cluster purity for every class present, which is defined as the fraction of documents in the clusters which agree with its dominant class. The average cluster purity over all clusters (weighted by cluster size) was reported as a surrogate for the quality of the clustering process.

Therefore, we have:

$$P=Ci/Ni \qquad (1)$$

where P is fraction of data points in cluster. The cluster purity lies between 0 and 1. Therefore a idle clustering will give a cluster purity of almost 1, and a poor clustering will give very low values of the purity.

Our TDT tracking system was influenced by prior work on detection. We found that the dual-threshold clustering paradigm of detection systems worked well for topic detection and tracking (after minimal modifications.) We found that many of the features that were beneficial for topic detection continued to be helpful for tracking. The basic document to document scoring formula was a symmetrised version of the PCA (Principle Component Analysis) formula.

For the browser update we use OSKM algorithm along with COATES and TDT. The OSKM algorithm identify and update the closest cluster center.

4. CONCLUSION

It's alternating minor iterations of content-based and auxiliary attribute-based clustering. We are going to test against a supervised clustering method which uses both text and side information. So improve the quality of the representation for clustering. Time delay will be reduced as the clusters are already stored in the database. So it will take less time to fetch the user query. Weight of content will be reduced. As the clusters are formed with the important side information and all the noisy data has been removed, the content weight is reduced.

REFERENCES

[1] C. C. Aggarwal and C.-X. Zhai, Mining Text Data. New York, NY, USA: Springer, 2012.

[2] C. C. Aggarwal and P. S. Yu, "A framework for clustering massive text and categorical data streams," in *Proc. SIAM Conf. Data Mining, 2006*, pp. 477–481.

[3] D. Cutting, D. Karger, J. Pedersen, and J. Tukey, "Scatter/Gather: A cluster-based approach to browsing large document collec- tions," in *Proc. ACM SIGIR Conf., New York, NY, USA*, 1992, pp. 318–329.

[4] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *in Proc. ACM SIGMOD Conf., New York, NY, USA,* 1996, pp. 103–114.

[5] Shi Zhong, "Efficient Online Sphercal K-means Clustering" in Proc. *International Joint Conference on Neural Networks, Montreal, Canada, jii1* -August4, 2005