Mining Text Data using different Text Clustering Techniques

Ratna S. Patil[#1], Prof. B. S. Chordia[*2]

[#]Master Student, SSVPS's BS Deore College of Engineering, Dhule, 424005, India
[*]Assistant Professor, SSVPS's BS Deore College of Engineering, Dhule, 424005, India

Abstract- *Text mining is referred as text data mining or knowledge discovery from textual databases. The organization of text is a natural practice of humans and a crucial task for today's vast databases. Clustering does this by assessing the similarity between texts and organizing them accordingly, grouping like ones together and separating those with different topics. Clusters provide a comprehensive logical structure that provides exploration, search and interpretation of current texts documents, as well as organization of future ones. Side information is available along with the text documents and may be of different kinds, which are embedded into the text document. However this side-information may be difficult to estimate. In such cases, it can be risky to include side-information into the mining process, because it can either increase the quality of the representation for the mining process. Therefore, so as to maximize the advantages from using this side information, to minimize the time complexity of clustering process and to remove impurity of clusters partition based text clustering techniques are used like k-means &k-Windows algorithm. Experimental results show that, K-Windows clustering technique is giving better results as compared to K-means clustering technique and also shows that side information is effectively used for mining the data.***

Key Words: *Clustering algorithms, Text Mining, Data Mining.*

I. INTRODUCTION

Mining is the process of inferring for patterns within a structured or unstructured data. There are various mining methods out of which they differ in the context and type of dataset that is applied. The process of extracting knowledge from unstructured text led to the need for various mining techniques for useful pattern discovery. Data Mining (DM) and Text Mining (TM) is similar in that both techniques "mine" large amounts of data, looking for meaningful patterns.Some of the mining types are data, text, web, business Process and service mining. DM is the process of retrieving information from large amounts of data to view the hidden knowledge and facilitate the use of it to the real time applications. DM consists of data analysis algorithms. Some techniques of Data Mining used for analysis are Clustering, Association, and Classification etc. Text mining is referred as text data mining or knowledge discovery from textual databases, it refers to the process of extracting interesting and non-trivial patterns or knowledge from text documents as shown in figure 1. TM starts with a collection of documents; which would retrieve a particular document and preprocess it by checking format and present text in proper format. Then it would go through a text analysis phase, sometimes repeating techniques until information is extracted. Three text analysis techniques are shown in figure 1, but many other combinations of techniques could be used depending on the goals of the organization. The following figure explores the detail processing methods in Text Mining.

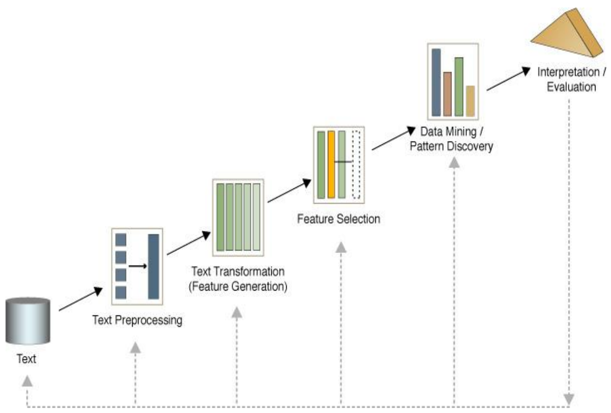
Text
Text Preprocessing
Text Transformation (Feature Generation)
Feature Selection
Data Mining / Pattern Discovery
Interpretation / Evaluation

Fig 1: Text Data Mining process

The problem of text clustering arises in the context of many application domains such as the web, social networks, and other collections. The rapidly increasing huge amounts of text data in the context of these large online collections has led to an interest in creating scalable and effective mining algorithms. The huge amount of work has been done in recent years on the problem of clustering in text collections in the database and information retrieval communities. However, this work is primarily designed for the various problems of pure text clustering, in the absence of other attributes. In many application domains, a tremendous amount of side information is also present along with the documents. This is because text documents occur in the context of a various applications in which there may be a large amount of other kinds of database attributes or Meta information which is useful for the clustering process. Examples of side-information are:

- In an application in which we track user access behavior of web documents, the user-access behavior is captured in the form of web logs. For each document, the auxiliary information may correspond to the browsing behavior of the different users. Such logs can be used to enhance the quality of the mining process and also application-sensitive. This is because the logs can often pick up subtle correlations in content, which cannot be picked up by the raw text alone [1].

- Many text documents contain links among them, which are also considered as attributes. Such links are useful for mining purposes.
- Many web documents have supporting information associated with them like the source of the document. In other cases, data such as ownership, location, or even temporary information may be informative for mining purposes. In many network and user-sharing applications, documents may be associated with user-tags, which may also be quite informative [1].

Side-information can sometimes be useful in improving the quality of the clustering process. Therefore, we will use an approach which carefully ascertains the correlation of the clustering characteristics of the side information along with the text content. This helps in purifying the clustering effects of both kinds of data.

Clustering is an effective technique for data analysis. Clustering is a widely studied data mining problem in the context of text domains. The problem finds various applications in customer segmentation, classification, collaborative filtering, visualization, document organization, and indexing [2]. Text Clustering is one of the most important research areas in the field of data mining. Data are grouped into clusters those having same data and those in other groups are dissimilar. It intends to decrease intra-class similarity while to increase interclass dissimilarity. Clustering is an unsupervised learning technique. Clustering is useful to obtain required patterns and structures from a large set of data. Clustering can be applied in many areas, such as marketing studies, DNA analyses, city planning, text mining and web documents classification.

Most existing methods of clustering can be categorized into: distance based clustering algorithms like agglomerative and hierarchical, Distance-based Partitioning Algorithms like k-means, K-medoid, K-Windows etc., A Hybrid Approach for clustering like scatter/gather technique etc. Partition based clustering generates a partition of the data such that objects in a cluster are more similar to each other than dissimilar objects in other clusters. Clustering is a technique to search hidden patterns from the existing datasets [3].In order to overcome the problems of pure text clustering for mining the data, side information avail with text data is used for mining the data. So, firstly preprocessing is applied on the dataset, and then distance measures are calculated. Text mining technique i.e. text clustering algorithms (k-means &k-Windows) is applied on similarity measures values. Then using side information results are evaluated.

II. RELATED WORK

Literature review in the area of mining indicates that there are several ways of mining text data so that efficient clusters should be formed and better results should be achieved. Database community has studied lots about the problem of text-clustering [2], [15] and [16]. In [15] they represent the novel algorithm termed as CURE which is more robust to outliers, and identifies clusters having non spherical shape and variance in size. In [16] proposed a method termed as BIRCH, which demonstrate especially for very large databases. Scalable clustering of multidimensional data of different types is discussed in [2], [15], and [16].

D.Cutting, D. Karger, J. Pedersen, and J. Tukey 1992 [3] explains the Scatter/Gather method which demonstrates that document clustering can be effective information access tool in its own right. They presented a document browsing technique that employs document clustering as its primary operation, they also presented fast clustering algorithms that support this interactive browsing paradigm. It uses a combination of agglomerative and partition based clustering.

Matrix-factorization techniques for text clustering are stated in [17]. In this technique words from the document based on their relevance to the clustering process are selected and to refine the clusters an iterative EM method is used.

R. Angelova and S. Siersdorfer 2006 [5] focus towards the problem of automatically structuring linked documents by using clustering. In contrast to traditional clustering, they studied the clustering problem in the light of available link structure information for the data set (e.g., hyperlinks among web documents). Their approach was based on iterative relaxation of cluster assignments, and which could be built on top of any clustering algorithm. That technique results in higher cluster purity, better overall accuracy, and made self-organization more robust.

The methods discussed in the above are focuses on the pure text data, these methods does not work for the text data which united with the other form of data. So, Charu C. Aggarwal, Yuchen Zhao and Philip S. Yu 2014 [1] designed an algorithm which combines classical partitioning algorithms with probabilistic models in order to create an effective clustering approach. They then show how to extend the approach to the classification problem. They presented methods for mining text data with the use of side-information. Many forms of text databases contain a large amount of side-information can be used in order to improve the clustering process. In order to design the clustering method, they combined an iterative partitioning technique with a probability approach which computes the importance of different kinds of side-information. This general approach is used in order to design both clustering and classification algorithms. They presented results on real data set which shows the effectiveness of their approach. The results showed that the use of side-information can increase the quality of text clustering and classification, while maintaining a high level of efficiency. They have used k-means clustering technique. k-means is computationally very expensive for the very large sets of patterns met in real life applications. On the other hand, k-means often converges to a local minimum.

M. N. Vrahatis, B. Boutsinas, P. Alevizos, and G. Pavlides 2002 [13] has presented an improvement of the k-means clustering algorithm, aiming at a better time complexity and partitioning accuracy. This approach reduces the number of patterns that need to be examined for similarity, in each iteration, using a windowing technique. The latter is based on well-known spatial data structures, namely the range tree, which allows fast range searches.

Bentley, J. L. (1975) has developed the multidimensional binary search tree (or k-d tree, where k is the dimensionality of the search space) as a data structure for storage of information to be retrieved by associative searches. The k-d tree is shown to be quite efficient in its storage requirements.

III. METHODOLOGY

Figure 2 shows the general architecture of system model. The dataset is an unstructured dataset of documents which are pre-processed using the following three rules: 1) Tokenize the file into individual tokens using space as the delimiter. 2) Removing the stop word which does not convey any meaning. 3) Use porter stemmer algorithm to stem the words with common root word. Stop Word Removal: Sometimes a very common word, which would appear to be of little beneficial in helping to select documents matching user's need, is completely excluded from the selected documents. These words are treated as "stop words" and this technique is called stop word removal.

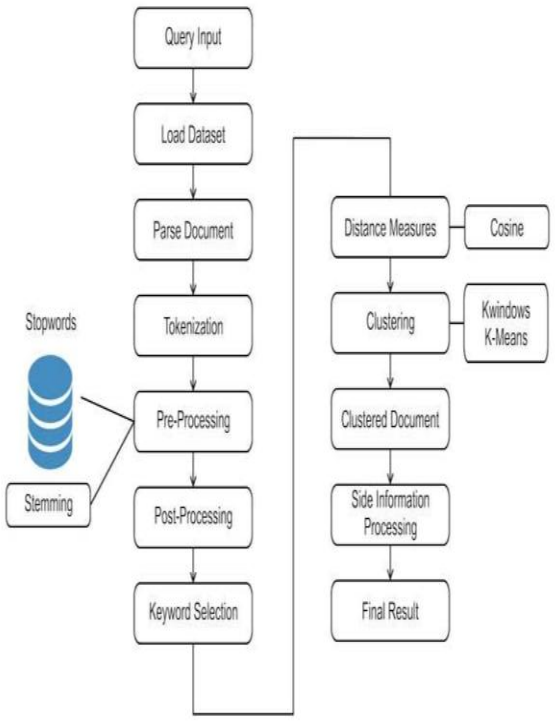

Fig 2: Working of Text Clustering System model

The general strategy for determining a "stop list" is to sort the terms by collectionfrequency and then to make the most frequently used terms are treated as stop list, the members of which are discarded during indexing. Some of the examples of stop-word are: a, an, the, and, are, as, at, be, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with etc. Here the input stop word file contains 641 words.

Stemming: Stemming is an analytical process in which the end of the words or the affixes of the derivational words are truncated to receive the base form of the word. Here, porter stemmer is used.

Then distance measure like cosine similarity is applied to find the similarity between the documents using the formula given in equation (3.1).

Cosine measure: When the angle between the two vectors is a meaningful measure of their similarity, the normalized inner product may be an appropriate similarity measure.

$$\text{Similarity } S(d_i, d_j) = \cos(\theta) = \frac{d_i.d_j}{||d_i||.||d_j||} \quad (3.1)$$

Using this cosine similarity formula, similarity between every document in dataset with other documents is calculated [5]. Then partition based clustering algorithms (i.e. k-means &k-Windows) are applied on this cosine similarity. Here K-means take 1-dimensional input as cosine similarity of each document while k-Windows take 2-dimensional input as cosine similarity and document number. Both these algorithm are described in section IV.

IV. PARTITIONAL BASED CLUSTERING ALGORITHMS

There are several partitional based clustering algorithms available like k-means, k-medoids, CLARANS, k-Windows etc. Here we have used k-means and k-Windows for text clustering explained as follows:

1. K-means Algorithm

K-means is one of the simplest unsupervised learning algorithms to group similar data objects. It was developed by J. MacQueen (1967) and then by J. A. Hartigan and M. A. Wong around 1975 K-means forms clusters for n objects based on the attributes into k partitions where k<n [5]. The k-means is a very popular algorithm particularly suited for implementing the clustering process because of its ability to efficiently partition huge amounts of patterns. The latter is true even in the presence of noise. Although direct K-means is defined over numerical continuous data, it is the basic framework for defining variants capable of working on both numerical and categorical data. The K-means consists of two main phases. During the first phase, a partition of patterns, in k clusters is calculated, while during the second phase, the quality of the partition is determined. K-means is implemented by an iterative process that starts from a random initial partition. The latter is continually recalculated until its quality function reaches an optimum. In particular, the whole process is built upon four basic steps:

(1) Selection of the initial k centroid as a seed,

(2) Assignment of each pattern to a cluster with nearest mean or centroid,

(3) Recalculation of k centroids for clusters, and

(4) Computation of the quality function.

The steps 2, 3, 4 are performed iteratively until convergence. Most clustering algorithms which are variants of k-means have been proved convergent [5]. On the other hand, k-means-type algorithms often terminate at a local minimum. Formally, let $i_{1,} \dots i_n$ be the input patterns. Each of them is represented by a d-tuple $\{(an_1, av_1), \dots, (an_d, av_d)\}$ where an_j, av_j, $1 \leq j \leq d$ denote, respectively, the name and the value of the j^{th} numerical attribute, whose domain is the set of reals R. Let the k first means be initialized to one of n input patterns $i_{m1},\dots,i_{mk}$. These k means define the set C of clusters $C = \{C_j \mid 1 \leq j \leq k\}$. The goal of the algorithm is to minimize the following quality function:

$$E=\sum_{j=1}^{k} \sum_{i_l \in C_j} q(i_l, i_{mj}) \qquad (4.1)$$

In direct k-means q is defined by the squared Euclidean distance, thus $q(y, z) = \|y-z\|^2$, where $\|\cdot\|$determines the Euclidean norm. The k-means algorithm is computationally very expensive for large sets of patterns. It requires time proportional to the product of the number of patterns, the number of clusters and the number of iterations. More specifically, in the algorithm above, the first loop, for each iteration, has a time complexity O(ndk), the second O(nd) and the quality function is calculated in O(nd). Thus the whole algorithm has a time complexity O(ndkt), where t is the number of iterations [5]. Improvement of the computational complexity is achieved either by sophisticated initialization methods (e.g., [6, 7, 10]) or by reducing the number of (dis)similarity calculations (e.g., [8, 9, 11]). The k-Windows algorithm is based on the latter approach.

2.K-Windows Algorithm

K-Windows algorithm deals with this problem by using a windowing technique, which reduces significantly the number of patterns that need to be examined at each iteration. Moreover, the basic operation in the first loop is the assignment of patterns to clusters, by performing arithmetic comparison between two numbers. The key idea behind this technique is to use a window in order to determine a cluster. The window is explained as an orthogonal range in the d-dimensional Euclidean space, where d is the number of numerical attributes. Here, 2-dimensional data is used. The magnitude of A depends on the density of the data set. Which is define, across each different direction i,

$$A_i = \frac{(\text{mean distance among patterns in i})}{(\textit{number of windows})} \times 0.5 \qquad (4.2)$$

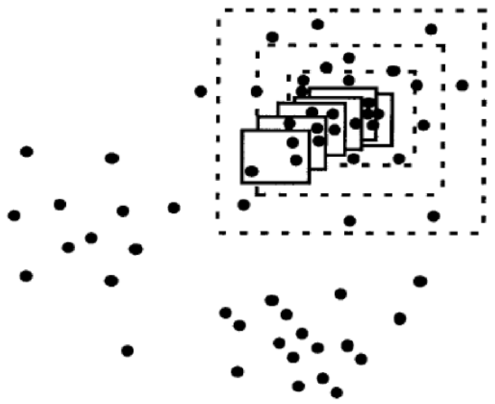

Fig 3: Movements and enlargements of a window.

Intuitively, we try to fill the mean space between two patterns with non-overlapping (thus we scale by 0.5) windows. Every pattern that lies within a window is treated as belonging to the corresponding cluster. Repeatedly, each window is moved in the Euclidean space by centering itself on the mean of the patterns included. This takes place till further movement results in an increase in the number of patterns that lie within it which is shown by solid lines in figure 3. After this step, we can determine the means of clusters as the means of the corresponding windows. However, since only a limited number of patterns are considered in each movement, the quality of a partition may not be optimum. The quality of a partition is calculated in a second phase. At first, we enlarge windows in order to contain as many patterns from the respective cluster. The quality of a partition is determined by the number of patterns contained in any window, with respect to all patterns.

The k-Windows clustering algorithm is as follows:

```
ALGORITHM  K-Windows.
**Input** k, A, t
**initialize** k means $j_{n1}$, ..., $j_{nk}$ along with their
k d-ranges $w_{n1}$, ..., $w_{nk}$ each of area A
**repeat**
        **for each** input pattern i, $1 \le p \le m$
        **do**
        **assign** $i_l$ to $w_i$ ,
        so that $i_l$ lies within $w_i$
**for each** d-range $w_i$
        **do**
        calculate its mean $i_{ni} = \frac{1}{|w_i|}\sum_{j_p \in w_i} j_p$
        (4.3)
        and recalculate d-ranges
**until** no any pattern has changed d-ranges
enlarge d-ranges up to no significant
change exists in their initial mean
**compute** the ratio $r = \frac{1}{n}\sum_{j=1}^{k} |i_p \in w_j|$
        (4.4)
**if** r < t
        **do** reexecute the algorithm
```

At first, k means are selected (possibly in a random way). Initial d-ranges (windows) have as centers these initial means and each one is of area a. Then, the patterns that lie within each d-range are found. As here the data is 2 dimensional, an orthogonal range search [11] is used. An orthogonal range search is based on a preprocess phase where a range tree i.e. here binary tree is constructed. Patterns that lie within a d-range can be found by traversing the binary tree, in polylogarithmic time. In the third step, the mean of patterns that are present within each range is calculated. Each such mean defines a new d-range that is considered a movement of the previous d-range. The last two steps are executed repeatedly, until no d-range includes a significant increment of patterns after a movement [13].

3. Kd Tree Search

Kd Tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g.range searches and nearest neighbor searches). k-d trees are a special case of binary space partitioning trees invented by Jon Louis Bentley in 1975 for multidimensional data. The *k*-d tree is a binary tree in which every node is a *k*-dimensional point. Every non-leaf node can be thought of as implicitly generating a splitting hyperplane that divides the space into two parts, known as half-spaces. Points to the left of this hyperplane are represented by the left subtree of that node and points right of the hyperplane are represented by the right subtree. The hyperplane direction is chosen in the following way: every node in the tree is associated with one of the k-dimensions, with the hyperplane perpendicular to that dimension's axis. So, for example, if for a particular split the "x" axis is chosen, all points in the subtree with a smaller "x" value than the node will appear in the left subtree and all points with larger "x" value will be in the right subtree. In such a case, the hyper plane would be set by the x-value of the point, and its normal would be the unit x-axis.

Let us consider the procedure for constructing the kd-tree.

It has two parameters, a set of points and an integer. The first parameter is set for which we want to build kd-tree, initially this the set S. The second parameter is the depth of the root of the sub tree. Initially the depth parameter is zero. The procedure returns the root of the kd-tree.

```
**Procedure name BUILDKDTREE(S,depth)**
**Input:** A set of points S and the current depth.
**Output:** The root of the kd-tree storing S.
**if** S contains only one point
        **then** return a leaf storing this point
**else if** depth is even
        **then** Split S into two subsets with a vertical line l through the median x-coordinate
        of the points in S.  $S_1$ be the set of points to the left of l or on l, and let $S_2$ be the
        set of points to the right of l.
**else** Split S into two subsets with a horizontal line l through the median y-coordinate
      of the points in S. Let $S_1$ be the set of points to the below of l or on l, and let $S_2$ be
      the set of points above l.
$v_{left}$←BUILDKDTREE($S_1$, depth +1).
$v_{right}$←BUILDKDTREE($S_2$, depth +1).
Create a node v storing l, make $v_{left}$ the left child of v, and make $v_{right}$ the right child of v.
return v.
```

As the kd-tree is binary tree. So, kd-tree for a set of n-points uses O(n) storage and and can be constructed in O(n logn) [11].

V. EXPERIMENTAL RESULTS

Experimental results have been evaluated on an Intel core2 DUO CPU with 2GBRAM under 64-bit Windows 8 operating system. This system is implemented in java using jdk 1.8. 20 NewsGroup Dataset [14] is used for experiments which a collection of approximately 20,000 newsgroup documents, partitioned among 20 different newsgroups like graphics, hardware, politics etc. It was originally collected by Ken Lang. Now a day, The 20 newsgroups collection has become a popular data set for experiments in text applications, such as text classification and text clustering.But this data may be noisy so data preprocessing is carried out to get the clean data. Data preprocessing consists various steps like tokenization, stop words removal, Stemming etc. We have used 641 stop words for stop word removal process and porter stemmer is used for stemming. Then vector space model is generated using tf-idf values. Using these vectors, cosine similarity is calculated between documents. This is given as input to the K Means & K Windows clustering. K Means works on single dimensional data. So, here it works on cosine similarity scores of documents. But K Windows works on 2 dimensional or multidimensional data. But, Data is 2 Dimensional. This is given as input to the K Windows i.e. (Cosine Similarity Scores, Document number). Both algorithms are tested for different size of cosine score values as 5, 10, 15, 20, 25, 30 and 35. According to that, it's running time in milliseconds, number of clusters generated by both algorithms & memory usage by both these algorithms in kilobytes is tabulated in the following Table 1. Using this information, graphs are generated which shows the comparison between K Means & K Windows algorithm.

Table I: Results of K Means & K Windows

Cosine Similarity Scores(Data Size)	Running Time (in Milliseconds)		Number of clusters generated		Memory Utilized in KB	
	k-Means	k-Windows	k-Means	k-Windows	k-Means	k-Windows
5	7	1	2	2	1587	848
10	9	1	3	4	1959	696
15	9	2	3	7	1778	682
20	9	2	3	9	1985	533
25	9	2	3	11	1972	682
30	9	2	3	13	1978	696
35	10	3	3	14	1962	848

Figure below shows experimental results on dataset of different data size as no. of cosine scores applied on both K Means & K Windows.

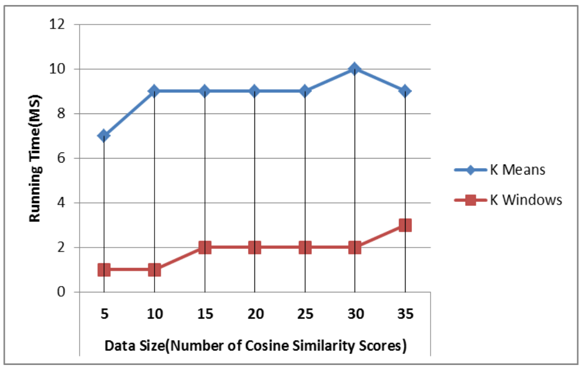

Fig 2:Comparison of running time required vs. data size for both algorithms

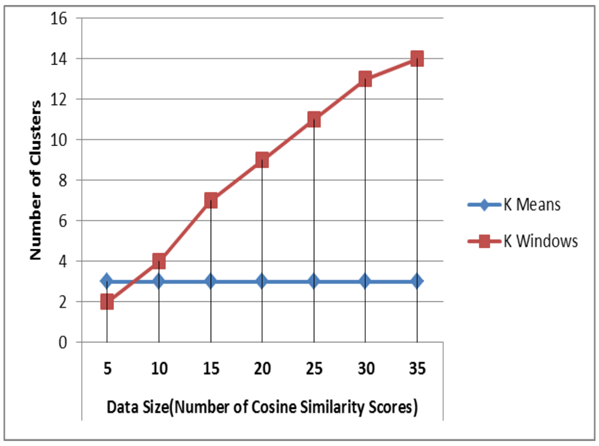

Fig 3: Comparison of number of cluster generated vs. data size for both algorithms

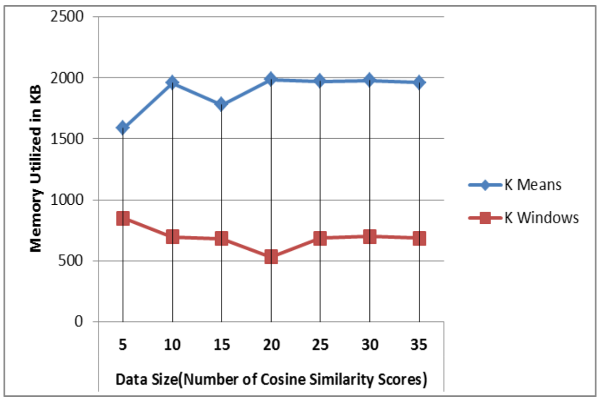

Fig 4: Comparison of memory utilized vs. data size

for both algorithms

The quality of a clustering result is evaluatedusing evaluation measure like purity is widely used to evaluate the performance of unsupervisedlearning algorithms [1].To begin with, each cluster is labeled with the majority category that appears in that cluster. Moreover, if a categorylabel has been assigned to a cluster, it still can be assigned toother clusters if it is the dominant category in that cluster.Based on the cluster labels, the purity and entropy measures are computed as follows. Thepuritymeasure evaluates the coherence of a cluster, thatis, the degree to which a cluster contains documents from asingle category. Given a particular cluster$C_i$of sizeni, thepurity of$C_i$is formally defined as

$$P(C_i) = \frac{1}{n_i} max_h (n_i^h) \qquad (5.1)$$

Where, $max_h (n_i^h)$ is the number of documents that are fromthe dominant category in cluster$C_i$andn_i^hrepresents thenumber of documents from cluster$C_i$assigned to categoryh. Purity can be interpreted as the classification rate underthe assumption that all samples of the cluster are predictedto be members of the actual dominant class for the cluster.For an ideal cluster, which only contains documents from asingle category, its purity value is 1. In general, the higherthe purity value, the better the quality of the cluster is[1].

Here, overall purity of clusters for both algorithms is listed in Table II and which graphically in fig 5.

Table II: Results of Purity Of Clusters of both k-Means & K-windows Algorithm

	K-Means	**K-windows**
Overall Purity of clusters	0.044	0.4145

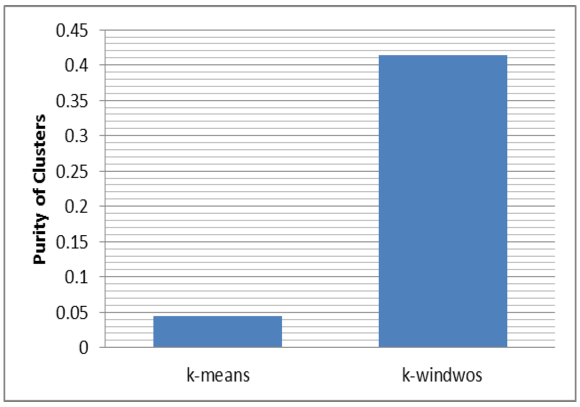

Fig 5: Cluster Purity Comparison between k-Means & K-windows

VI. CONCLUSION

Text mining is performed using side information and clustering is performed using K-mean &K-Windows. Many forms of text data is gathered from databases contain a large amount of side information or meta information, which may be used in order to improve the quality of clustering results. Experimental results show that use of side information can enhance the quality of text clustering and performance is evaluated in terms of memory utilization & running time. K Windows clustering technique is efficient as compared to K Means. So, the quality of searching the query is enhanced using side information.

REFERENCES

[1] Charu C. Aggarwal, Yuchen Zhao and Philip S. Yu ,"On the Use of Side Information for Mining Text Data", IEEE transactions on knowledge and data engineering, vol. 26, no. 6, June 2014
[2] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in Proc. ACM SIGMOD Conf., New York, NY, USA, 1998, pp. 73–84.
[3] D. Cutting, D. Karger, J. Pedersen, and J. Tukey, "Scatter/Gather:A cluster-based approach to browsing large document collections," in Proc. ACM SIGIR Conf., New York, NY, USA, 1992, pp. 318–329.
[4] C.C. Aggarwal and P.S.Yu,"On text clustering with side information," in Proc. IEEE ICDE Conf., Washington, DC, USA, 2012.
[5] Lior Rokach, Oded Maimon, "Chapter 15 Clustering Methods", data mining and knowledge discovery handbook.
[6] P. S. Bradley and U. M. Fayyad, Refining initial points for k-means clustering, in ''Proceedings of the IJCAI-93, San Mateo, CA,'' pp. 1058–1063, 1983.
[7] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Mining Knowledge Discovery 2 (1998), 283–304.
[8] A. K. Jain and R. C. Dubes, ''Algorithms for Clustering Data,'' Prentice–Hall, Englewoods Cliffs, NJ, 1988.
[9] D. Judd, P. McKinley, and A. Jain, Large-scale parallel data clustering, in ''Proceedings of Int. Conference on Pattern Recognition,'' 1996.
[10] C. Pizzuti, D. Talia, and G. Vonella, A divisive initialization method for clustering algorithms, in ''Proc. PKDD 99—Third Europ. Conf. on Principles and Practice of Data Mining and Knowledge Discovery,'' Lecture Notes in Artificial Intelligence, Vol. 1704, pp. 484–491, Springer-Verlag, Prague, 1999.
[11] Bentley, J. L. (1975). "Multidimensional binary search trees used for associative searching". *Communications of the ACM*. 18 (9): 509.
[12] C. C. Aggarwal and C.-X. Zhai, "A survey of text classification algorithms," in Mining Text Data. New York, NY, USA: Springer, 2012.
[13] M. N. Vrahatis, B. Boutsinas, P. Alevizos, and G. Pavlides, The New k-Windows Algorithm for Improving the k-Means Clustering Algorithm, journal of complexity 18, 375–391 (2002).
[14] http://qwone.com/~jason/20Newsgroups/
[15] R. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in Proc. VLDB Conf., San Francisco, CA, USA, 1994, pp. 144–155.
[16] 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.
[17] W. Xu, X. Liu, and Y. Gong, "Document clustering based on nonnegative matrix factorization," in Proc. ACM SIGIR Conf., New York, NY, USA, 2003, pp. 267–273.