IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-9 E-ISSN: 2347-2693

Evaluation of Clustering Algorithm in Data Mining

Arpit Agrawal

Institute of Engineering and Technology, Devi Ahilya University Indore, Madhya Pradesh

Available online at: www.ijcseonline.org

Received: 12/Aug/2017, Revised: 25/Aug/2017, Accepted: 17/Sep/2017, Published: 30/Sep/2017

Abstract- Text mining is the use of data mining techniques to unstructured text in order to extract important and nontrivial knowledge. One of the key methods of text mining, or the unsupervised classification of related content into various categories, is text clustering. The performance of text clustering is being improved in this study. We looked on four areas of the text clustering algorithms: document representation, document similarity analysis, high dimension reduction, and parallelization. We suggest a collection of very effective text clustering techniques that focus on the special features of unstructured text databases. All of the suggested algorithms have undergone thorough performance studies. We contrasted these techniques with current text clustering algorithms in order to assess their performance.

Keyword- Cluster Algorithm, Data Mining, bisecting k-means, FIHC, CFWS find CFWMS

I. INTRODUCTION

The performance of text clustering is being improved in this study. We looked on four areas of the text clustering algorithms: document representation, document similarity analysis, high dimension reduction, and parallelization. We suggest a collection of very effective text clustering techniques that focus on the special features of unstructured text databases.

Two novel text clustering techniques are first suggested. We employ a document representation that preserves the sequential link between words in the documents, in contrast to the vector space model, which sees documents as a collection of words. In these two techniques, the size of the database is condensed by taking into account common word sequences, and the similarity of two documents is determined by their shared use of common word sequences. The second step is the proposal of a text clustering algorithm with feature selection. This approach does feature selection during clustering to gradually reduce the high dimension of the database. A novel statistical dataset that can quantify both positive and negative term-category dependence is used in the new feature selection methodology.

Third, a collection of novel text clustering methods built on the k-means algorithm is created. A new function involving global information should be used in place of the cosine function. The recently created neighbour matrix is used by this new function. The suggested algorithms incorporate a new technique for choosing initial centroids and a new heuristic function for choosing a cluster to split. The message-passing multiprocessor systems are given a new parallel approach for bisecting k-means.

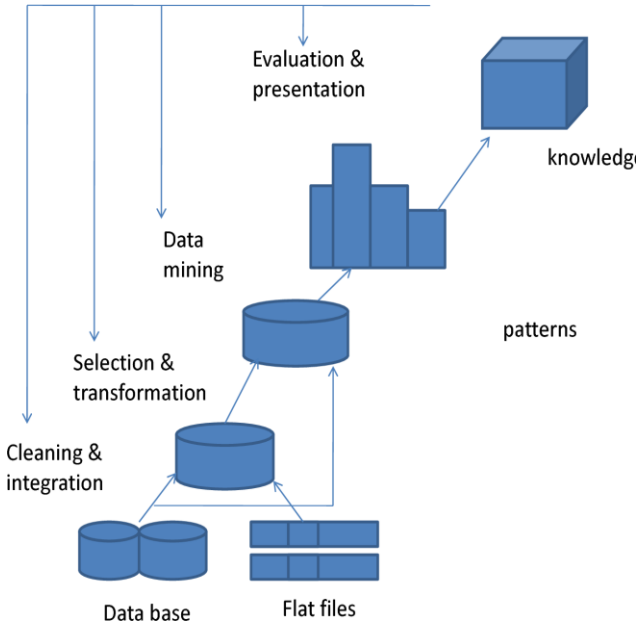

Fig.1 Clustering in data mining

Theorem 1: If a word wi is a member of a common k-word sequence, it must also be a member of a frequent k-word set. A member of a frequent k-word set, on the other hand, is not always a member of a frequent k-word sequence, where the order of the k words is important. This is quite simple.

Theorem 2: All of a k-word sequence's k1-word subsequences are common if a k-word sequence is frequent. From the definition of the subsequent, it is simple to demonstrate.

This research offered a way for enhancing text clustering performance. The text clustering algorithms were looked into.

II. BACKGROUND TECHNIQUES

Frequent Word Sequences

As a result, a text document d may be represented as d = w1; w2; w3;::: > in our method, where w1; w2; w3;::: are words that exist in d. A word set is frequent if its support is at least the user-specified minimum support, much like a frequent itemset in the association rule mining of a transaction data set [1] is frequent. This indicates that this word set is present in at least the required minimum number (or proportion) of texts. A common word set with k words in it is known as a frequent k-word set.

An ordered sequence of two or more words is called a word sequence. The symbol for S is w1; w2;::: >. In this chapter, the letters F S stand for a common word combination. For instance, F S = w1; w2; w3; w4 >, where w2 may not always come just after w1 in a text page. As long as w2 comes after w1 and there aren't many words in between, there may be words. If these four words (w1; w2; w3; and w4) exist in the text document d in the designated order, then the word sequence is supported. When there are at least the required minimum number (or percentage) of documents supporting a word sequence S, it is a F S. Sequences that appear more than once in the same document are considered as one instance. As a result, our definition of common word sequence can better accommodate the differences across human languages.

Finding Frequent 2-Word Sets

In this stage, words that are insufficiently frequent to be included in a frequent k-word sequence for k 2 are eliminated with the intention of lowering the dimension of the database (i.e., the number of unique words). This process is easy and clear-cut. We find the common 2-word sets that meet the minimal support using an association rule miner. A set WS is created that contains all of the words in common 2-word sets. We know that elements of the frequent word sequences of all lengths k, k 2 must be included in the set WS based on Theorems 1 and 2.

We eliminate all terms from the papers that are not found in WS after identifying the frequently occurring 2-word groupings. The documents that remain after the deletion are referred to as compact documents.

Building a Generalized Suffix Tree (GST)

Our objective is to identify the database's most common word sequences. To identify all common word sequences, we use the suffix tree, a well-known data structure for sequence pattern matching. Each compact document is viewed as a collection of words that are individually added to a generalised suffix tree (GST). Finally, we can discover all of the common word sequences in this database by compiling the data kept in all the nodes of this GST.

In reality, a compressed suffix tree for a string S is a list of S's non-empty suffixes. The suffixes of a group of strings are combined into a GST, which is a suffix tree. In our situation, we create a GST out of all the compact documents in the text database, therefore changes are made to the GST's structure to suit our requirements. The words "suffix tree" and "GST" will be used interchangeably throughout the remainder of this chapter.

A directed, rooted tree is what a suffix tree is. Internal nodes and suffix nodes are the two different types of nodes. There are at least two children in every internal node. Each edge is identified as l and is labelled with a non-empty substring of the string S. The labels along the path from the root to a given node, n, are concatenated to form the node's label. StringL of n, or simply n:stringL, is used to express this label. Different edges from the same node must have labels with unique starting words. There is a suffix node with the label s for each suffix s in the string S. Each suffix node has a collection of document ids linked with it. If a substring of a document ends at a node, then the document id is inserted into the id set of the node.

These document ids are used to check the multiple occurrences of a sequence in the same document. Node 2's word sequence is a subsequence of node 1's word sequence, while node 1's id set is a superset of node 2's. Some of the information would be lost if we merely found the most frequent word sequences. The most common word combinations can serve as a document's content description. However, a common word combination "boys play" is the best way to sum up the information in this sample database. Young boys play is a maximum frequent word sequence that only covers the first two documents, d1 and d2, and boys play basketball is another maximal frequent word sequence that only covers the first two documents, d1 and d3.

Finding all the common word sequences may reveal certain instances of duplication data, such as the sequences of nodes 6 and 7 in this illustration. If space needs to be saved, we may merge these two nodes into one without losing any information by matching their document id sets and word sequence members. By identifying all common word sequences, as demonstrated by this example, we may get some helpful data about the database to use in future information retrieval and data mining activities.

III. PROPOSED TECHNIQUES

The performance of text clustering is being improved in this study. We looked on four areas of the text clustering

algorithms: document representation, document similarity analysis, high dimension reduction, and parallelization.

We employ a document representation that preserves the sequential link between words in the documents, in contrast to the vector space model, which sees documents as a collection of words. In these two techniques, the size of the database is condensed by taking into account common word sequences, and the similarity of two texts is determined by their shared use of common word sequences.

Finding the frequent word sequences has two steps: finding frequent 2-word sets first, then finding frequent word sequences of all length.

Clustering Based on Frequent Word Sequences (CFWS) Algorithm

Our CFWS algorithm has two steps: building a GST to find frequent word sequences, then combining clustering candidates to obtain the final clustering.

Finding Frequent Word Sequences and Collecting the Cluster Candidates

The technique is used to create a GST for the database. The minimal support of frequent word sequences is often in the range; if it were too big, the total number of frequent words would be quite little, leaving insufficient information about the original data set in the compact texts that resulted. Many documents won't be analysed in this situation since they don't support any common words, and the final clustering result won't include those papers.

We may utilise these sets directly to acquire the cluster candidates because the document id sets are kept at the GST's suffix nodes. The cluster candidates can only be produced by the tree nodes that represent frequently occurring word sequences.

IV. COMBINING THE CLUSTER CANDIDATES

The cluster candidate's F S provides a summary of the contender. For instance, we know that cc1 includes the papers pertaining to the game the boys play as F S1 = "boys play." But occasionally, we don't require such clusters; instead, we can be interested in a broader broad subject, like the popular sports among teens. The relevant cluster candidates can then be combined in such scenario. Two clusters that are strongly connected to one another are also combined using the agglomerative clustering technique. We employ the k-mismatch notion of sequential patterns instead of a distance function to gauge the proximity of two cluster possibilities.

A k-mismatch of p is a jpj-substring of t that matches (jpj k) characters of p given a pattern p, a text t, and an independent of the lengths of p and t xed integer k. It therefore compares p and k mismatches. By creating the GST of the document collection, we are in the process of comparing the differences between the frequent word sequences that were discovered. Insertion, deletion, and replacement are the three different sorts of mismatches that can occur between the frequently occurring word sequences F Si and F Sj. The larger pattern F Sj is created by inserting k words into the shorter pattern F Si. This process is known as insertion.

The lengthier pattern F Sj is reduced to the shorter pattern F Si by eliminating k words from it. The link between two patterns of equal length, F Si and F Sj, is known as substitution. For example, F Si becomes F Sj by replacing k words.

We combine those cluster possibilities with k-mismatched patterns (frequent word sequences) for a given k. Two cluster candidates with k-mismatched patterns are thought to cover related themes. For instance, we may state that the subject matter covered by the cluster candidate i with F Si = "boys play" is similar to the subject matter covered by the cluster candidate j with F Sj = "play basketball". Therefore, we may combine them to create a larger cluster that addresses the question of who plays which game. How ne the overall clustering would be depends on the parameter k's value. The nal clusters, which are the nest clusters that may be discovered from the GST, are the cluster candidates if k is 0. As k value increases, the topic covered by each cluster of the nal clustering would be more general.

Some clusters may have excessive document id set overlap when we combine numerous cluster candidates into clusters. The range of is obviously [0; 1]: When = 1, these two clusters share the same collection of documents, however this does not imply that these two clusters are similar because this set of documents may cover two different themes. When = 0, these two clusters are disjunct.

Finally, as they lack a common word sequence, we gather the papers that do not belong to any cluster. These texts collectively constitute a cluster, and "other issues" might be used to describe their subject.

Our text clustering algorithm CFWS is summarized as follows:

a) Given a collection of text documents D = fd$_1$; d$_2$; d$_3$; : : : d$_n$g, nd the set of frequent 2-word sets of D with the user-specified minimum support. Obtain WS, the set of all
b) words, each of which is a member of a frequent 2-word set.
c) Reduce each document d$_i$, 1 i n, into a compact

document d^0_i by removing every word w from d_i if w 2= WS.
d) Insert each compact document into the GST.
e) Using the depth- rst traversing, visit every node in the GST. If a node j has a frequent word sequence F S_j with the set of document ids Ids_j , create a cluster candidate- cc_j ⌈F S_j ; Ids_j ⌉.
f) Merge the related cluster candidates into clusters based on the k-mismatch concept.
g) Combine the overlapping clusters if necessary.

Clustering Based on Frequent Word Meaning Sequences (CFWMS)

We initially count the number of times each word appears in the texts before using the CFWS method to locate common word sequences in the text database. A literal term in text documents is referenced by a word form in this word form matching method. The lexicalized idea that a word form may be used to represent, on the other hand, is referred to as a word meaning [66]. Synonyms are words that have the same meaning but are used to convey it in different ways in the actual world. A set of word forms that are synonyms is known as a synonym set, or simply synset, and it may be used to describe the meaning of a word.

Word meanings convey a document's subject matter more effectively than word forms do. We provide a brand-new technique called Clustering based on Frequent Word Meaning Sequences (CFWMS), which employs frequent word meaning sequences as the evaluation of the proximity of texts in order to enhance the quality of clustering.

The word meanings that the word forms in the texts first express are translated in CFWMS. Each text document is then viewed as a series of word meanings after conversion. A text document d, for instance, can be seen as as d =< SS_1; SS_2; SS_3 ; : : : >. If more than a predetermined number (or percentage) of papers include a word meaning sequence, it is deemed common.

Finding the appropriate word meaning that a word form represents in a certain lexical context is not a simple challenge because most words have numerous word meanings. To anticipate the actual word meaning, our system uses a meaning union (MU), which is a union of synset connections. You can consider a synset connection to be its own meaning union.

For instance, there is a meaning union M U = fSS_1 ! SS_2 ; SS_3 ! SS_4g, where SS_1=fboxg, SS_2 =fcontainerg, SS_3=fbox, logeg, and SS_4 =fcompartmentg. We anticipate that the genuine word meaning conveyed by the word form "box" in this text is one of the synsets in M U.

We will describe how to find the meaning unions. In this manner, a document d can be represented by meaning unions as d^0 =< M U $_1$; M U $_2$; M U $_3$; : : : >. following conversion. Similarly, F M S =< M U $_1$; M U $_2$; : : : > might be used to express a common word meaning sequence.

CFWMS Algorithm

There are three stages in our CFWMS algorithm: Finding common word meaning sequences and gathering cluster candidates are the first two steps in preprocessing texts to create meaning unions from word forms. The third step is merging cluster candidates to create the final clusters.

Document Preprocessing Using Word-Net

The transformation of word forms into meaning unions using an ontology is the most crucial step in the preparation of texts. We used Word-Net, an online lexical reference system, as an ontology.

We remove stop words but do not stem when converting word forms into word meanings. Only nouns, verbs, adjectives, and adverbs are included in Word-Net. We concentrate just on nouns and verbs and eliminate all adjectives and adverbs from the texts since they are more crucial in reflecting the substance of documents and also primarily make up the frequent word meaning sequences. We retain the word forms in the papers that do not have entries in Word-Net since they may include particular information about the documents.

There must be two conversion passes for each document. For each word and verb in the text, the meaning union (MU) is first extracted from Word-Net. From the most to the least commonly used, Word-Net's synsets indicate a word form's several word meanings. We choose the first two synsets containing the word form for each noun and verb. One direct hypernym synset is also obtained for each synset that is chosen. If there is just one synset in the word form, one inherited hypernym synset is also retrieved. Every word form therefore has its own meaning union, which includes at least one synset connection.

For example, a document d_1 =< w_1; w_2; w_3 > can be converted to a sequence of meaning unions as d^0_1 =< M U $_1$; M U $_2$; M U $_3$ >, where M U $_1$ = fSS_4 ! SS_5 ; SS_6 ! SS_7 g, M U $_2$ = fSS_3 ! SS_8; SS_9 ! SS_{10} g, M U $_3$ = fSS_1 ! SS_5 ; SS_2 ! SS_3g.

We aim to lessen the amount of distinct meaning unions in the transformed document during the second pass. comparable word meanings may be expressed by comparable word forms in one text. As a result, if two or more meaning unions share a synset, we replace them with a single meaning union that has a merged synset connection. In this instance, the frequency of the synsets in the document determines the

order of replacement. To that end, each synset's occurrences in the document's meaning unions are counted, and a list of synsets and the meaning unions they support is produced.

Combining the Cluster Candidates

The themes covered by such papers are described by a cluster candidate's frequent word meaning sequences. The k-mismatch idea employed in the CFWS method is not necessary when combining the cluster candidates. The rationale is because since word meanings rather than word forms were employed, the cluster candidates that we discovered are sufficiently generic. As in CFWS, all that has to be done is to look at the overlap between cluster candidates.

V. EXPERIMENTAL ASSESSMENT

We assess the effectiveness of our text clustering method in terms of the accuracy of clustering and the scalability of finding common word sequences. On a SuSE Linux computer with a Celeron 500 MHz processor and 384 MB of RAM, we implemented our method in C++.

VI. DATA SETS

Two sets of data are used for the performance assessment. One group consists of typical text document sets, which are frequently seen in studies on text clustering. They differ in terms of document size, cluster size, number of classes, database dimension, and dispersion of documents. To evaluate how well our algorithms performed on common text texts, we selected this collection of data sets.

The test data sets' documents have all been pre-classified into one or more categories. The accuracy of each clustering method is assessed using this information, which is concealed throughout the clustering procedures.

Evaluation of the Finding Frequent Word Sequences

We assessed the scalability of our strategy for locating common word sequences. Finding frequently occurring 2-word sets is the first phase of the mining process, followed by creating and traversing the GST. In this approach, we can identify frequent 2-word sets using any frequent itemset mining technique. We used the Apriori method, the most representative frequent itemset mining algorithm, in our experiment. There have been several other frequent itemset mining techniques developed, however their effectiveness in locating frequent 2-itemsets isn't very different. The minimal support level affects Apriori's effectiveness. The runtime of Apriori grows when there are more frequent itemsets when the minimum support is reduced.

Comparison of CFWS and CFWMS with Other Algorithms

We also used bisecting k-means and FIHC on the identical data sets to compare them to our CFWS and CFWMS. In comparison to k-means and agglomerative hierarchical clustering algorithms, bisecting k-means has been shown to consistently yield a better grouping outcome. Because FIHC employs frequent word sets like CFWS, it was chosen. We did not independently implement the bisecting k-means and FIHC algorithms in order to conduct a fair comparison. Before they were employed in our experiments, data sets underwent preprocessing. First, the stop words must be taken out of the texts. The words are then stemmed using the Porter's su x-stripping method [59] for the CFWS, bisecting k-means, and FIHC algorithms. Stemming is crucial since it may get rid of the little differences between words with the same meaning. We transformed word forms into word meanings for CFWMS using the WordNet ontology. As a result, the text database's dimension is further shrunk.

Table 1 shows the F-measures of four algorithms: bisecting k-means, FIHC, CFWS find CFWMS.

Table 1: F-measures of the clustering algorithms

Data Set	Bisecting k-means	FIHC	CFWS	CFWMS
Re1	0.606	0.506	0.651	0.721
Re2	0.677	0.74	0.79	0.79
Re3	0.675	0.39	0.703	0.701
Ce1	0.43	0.506	0.541	0.55
Ce2	0.466	0.46	0.48	0.48
Ce3	0.489	0.515	0.54	0.604
Se1	0.611	0.664	0.705	0.711
Se2	0.529	0.461	0.689	0.71
Se3	0.716	0.759	0.8	0.806

The purity values for the four clustering techniques are displayed in Table 2.

Table 2: Purity values of the clustering algorithms

Data Set	Bisecting k-means	FIHC	CFWS	CFWMS
Re1	0.622	0.527	0.749	0.751
Re2	0.796	0.652	0.679	0.81
Re3	0.777	0.444	0.504	0.675
Ce1	0.64	0.605	0.62	0.635
Ce2	0.69	0.585	0.601	0.637
Ce3	0.79	0.692	0.702	0.775
Se1	0.62	0.66	0.703	0.803
Se2	0.695	0.563	0.714	0.789
Se3	0.837	0.84	0.851	0.852
Se3	0.837	0.84	0.851	0.852

In the CFWMS, the degree of similarity across documents is determined by the frequency of word meaning sequences. Our CFWMS algorithm effectively captures the word meanings given by various word forms. Additionally, the common word meaning sequences serve as a representation of the themes addressed in ordinary text publications.

CFWS outperforms bisecting k-means and FIHC for the search query results (Se1, Se2, and Se3). The performance outcomes can be explained by the special qualities of certain document sets. These document sets were found in a search engine's retrieval lists for user queries. For each query, we carried out the retrieval using the Lemur Toolkit's straightforward TFIDF retrieval model. Compared to conventional text documents, the documents produced from this retrieval technique are more likely to share words. In other words, compared to ordinary document sets, the subjects covered by this kind of document sets are significantly more closely related. Our CFWS method can perform better for these kinds of data sets since it can cluster the documents into much smaller groups. The rationale is because by identifying common word sequences in the papers, our computer can detect subtle changes across subtopics. Our CFWS technique helps online search engines to give users more precise search results by grouping the retrieved content. Purity data demonstrate that CFWMS may enhance the clustering quality for both common text documents and search query results.

VII. CONCLUSION

CFWS, which stands for Clustering based on Frequent Word Sequences, is a brand-new text document clustering technique that we initially introduced in this paper. Unlike the conventional vector space model, our approach makes use of the document's word order. The themes addressed by the documents may be very effectively represented by frequent word sequences found from the document collection, and documents with the same frequent word sequences are grouped together in our technique. Next, we presented CFWMS, which stands for Clustering based on Frequent Word Meaning Sequences, as a brand-new text document clustering technique. Utilising common word meaning sequences to gauge how closely related papers are to one another, CFWMS improved CFWS. More accurately than frequent word sequences, frequent word meaning sequences can capture the subjects of texts. We preprocessed the papers using the WordNet ontology's synonyms, hyponyms, and hypernyms in order to identify common word meaning sequences. On the majority of our test data sets, CFWMS is more accurate than CFWS. According to the findings of our experiments, CFWS outperforms alternative clustering algorithms in terms of accuracy, particularly when it comes to the ne clustering of documents belonging to the same category, which is a highly helpful characteristic for contemporary online search engines.

VIII. REFERENCES

[1] Mohammed J. Zaki. Scalable algorithms for association mining. IEEE Trans. on Knowl. and Data Eng., 12(3):pp 372–390, 2000.

[2] P. Agrawal, O. Benjelloun, A. Das Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. "Trio: A system for data, uncertainty, and lineage". In Proc. Int. Conf. on Very Large Databases, 2006.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proc. ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, MN, 1994.

[4] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Züfle. Probabilistic frequent itemset mining in uncertain databases. In In Proc. 15th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, Paris, France, 2009.

[5] C. C. Aggrawal and P. S. Yu, \Finding Generalized Projected Clusters in High Dimensional Spaces," Proc. of ACM SIGMOD Int'l Conf. on Management of Data, 2000, pp. 70{81.

[6] J. Allan, \HARD Track Overview in TREC 2003 High Accuracy Retrieval from Documents,"Proc. of the 12th Text Retrieval Conference, 2003, pp. 24{37.

[7] M. R. Anderberg, Cluster Analysis for Applications, Academic Press, 1973.

[8] DPVG06] Nele Dexters, Paul W. Purdom, and Dirk Van Gucht. A probability analysis for candidate-based frequent itemset algorithms. In SAC ’06: Proceedings of the 2006 ACM symposium on Applied computing, New York, NY, USA, 2006. ACM, pp541–545.

[9] Gregory Buehrer, Srinivasan Parthasarathy, and Amol Ghoting. Out-ofcore frequent pattern mining on a commodity pc. In KDD ’06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, 2006, pp 86–95.

[10] Toon Calders. Deducing bounds on the frequency of itemsets. In EDBT Workshop DTDM Database Techniques in Data Mining, 2002.

