Data Mining Project

Documents Clustering using Association Rules

Presented to: **Dr. Walaa Khaled**

Submitted by: **Fatma El-Nagar**
Doaa' Ahmed

Date: **May 21st 2015**

Abstract: Document clustering is one of the important research issues in the field of text mining, where the documents are grouped without predefined categories or labels. High dimensionality is a major challenge in document clustering. Some of the recent algorithms address this problem by using frequent term sets for clustering. This report introduces a new methodology for document clustering based on Association Rules Mining. Our approach consists of three phases: the text preprocessing phase, the association rule mining phase, and the document clustering phase. The generated association rules are used for obtaining the partition, and grouping the partition that have the same documents.

1. Introduction

Document clustering is the task of automatically organizing text documents into meaningful clusters or group. Document clustering has been studied intensively because of its wide applicability in areas such as Web mining, Search Engines, Information Retrieval, and Topological Analysis. Unlike in document classification, in document clustering no labeled documents are provided. The problem of document clustering is generally defined as follows: Given a set of documents, partition them into a predetermined or an automatically derived number of clusters, such that the documents assigned to each cluster are more similar to each other than the documents assigned to different clusters. A major characteristic of document clustering algorithms is the high dimensionality of the feature space, which imposes a big challenge to the performance of clustering algorithms. They could not work efficiently in high dimensional feature spaces due to the inherent sparseness of the data. Next challenge is that not all features are important for document clustering, some of the features may be redundant or irrelevant and some may even misguide the clustering result.

In this report, we introduce our approach for document clustering based on strong association rules between the frequent term sets.

2. Consequently the Direct Hashing and Pruning algorithm in the mining process is presented. The Direct Hashing and Pruning (DHP) algorithm is in fact, a variation of Apriori algorithm. Both algorithms generate candidate k+1-itemsets from large k-itemsets, and large k+1-itemsets are found by counting the occurrences of candidate k+1-itemsets in the database. The difference of the DHP algorithm is that, it uses a hashing technique to filter out unnecessary itemsets for the generation of the next set of candidate itemsets.[3][4][6]
   The generated association rules are used for obtaining the partition, and grouping the partition that have the same documents. Furthermore, the resultant clusters are effectively obtained by grouping the partition by means of derived keywords. The rest of this paper is organized as follows: Section 2 presents the methodology for document clustering. The experimental results and post processing are presented in Section 3, and Section 4 outlines the attached source code description. [1]

3. Proposed System

The proposed approach for clustering document based on strong association rules is shown in Fig.1. The main characteristic of the approach is that it introduces an approach for document clustering based on strong association rules between frequent termsets. Our system consists of three phases: 1) Text Preprocessing Phase includes: filtration, stemming, and indexing of documents, 2) Association Rule Mining Phase using the Direct Hashing and Pruning (DHP) algorithm, and 3) Document Clustering Phase includes two main steps : the partitioning based on strong association rules and the clustering from the partitions. [2]

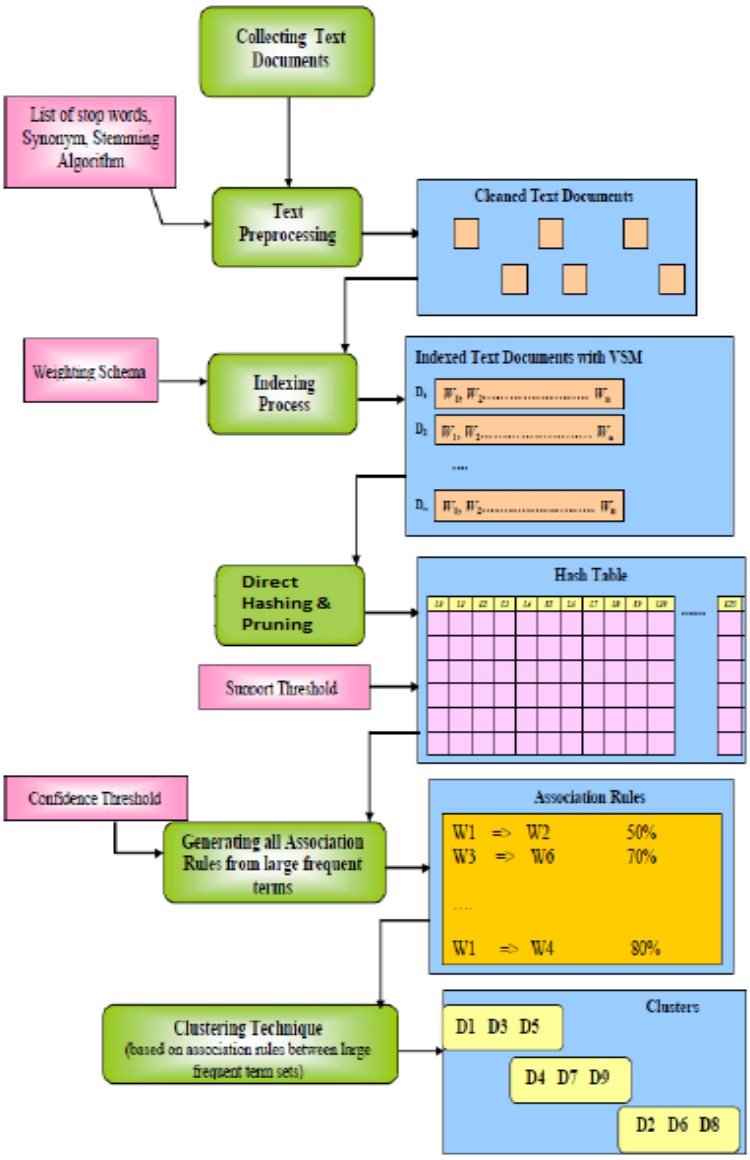


Fig.1 The methodology for document clustering

2.1 Text Preprocessing phase

The text preprocessing phase begins after collecting the text documents that need to be clustered. The documents are filtered to eliminate unimportant words by using a list of stopwords and eliminate strange characters that are non-letters and non-digits . After the filtration process, the system does word stemming that removes prefixes and suffixes of each word. Finally, the documents are automatically indexed based on the important words by using a weighting scheme.

2.1.1 Filtration

In this process, the documents are filtered by removing the unimportant word from documents content. Therefore, the unimportant words (noise words) get discarded or ignored (e.g. articles, pronouns, determiners, prepositions, conjunctions and common adverbs.

2.1.2 Stemming

Stemming is a technique that is commonly used to control the list of indexed words by removing the prefixes and suffixes of each word. The stemmer reduces similar words to the same root and this has two positive effects: 1) the number of indexed terms is reduced because similar terms are mapped into one entry. 2) The relevancy of the results is often improved by a significant margin.

2.1.3 Indexing

Our approach automatically indexes documents by labeling each document by a set of the most important words with their frequencies. The techniques for automated production of indexes associated with documents usually rely on frequency-based weighting schema. The weighting schema is used to index documents and to select the most important words in all document collections. The purpose of weighting schema is to reduce the relative importance of high frequency terms while giving a higher weight value for words that distinguish the documents in a collection. The weighting scheme TF-IDF (Term Frequency, Inverse Document Frequency) is used to assign higher weights to distinguished terms in a document. The weighting

scheme includes the intuitive presumption that is: the more often a term occurs in a document, the more representative of the content of the document (term frequency) but the more documents the term occurs in, the less discriminating it is (inverse document frequency).

$$w(i,j) = tfidf(d_i, t_j) = \begin{cases} Nd_i, t_j * \log_2 \frac{|C|}{Nt_j} & if\ Nd_i, t_j \geq 1 \\ 0 & if\ Nd_i, t_j = 0 \end{cases} \quad (1)$$

where *w(i,j) ≥0*, *Ndi,t j* denotes the number the term *t j* occurs in the document *di* (term frequency factor), *Nt j* denotes the number of documents in collection C in which t_j occurs at least once (document frequency of the term t_j) and $|C|$ denotes the number of the documents in collection C. The first clause applies for words occurring in the document, whereas for words that do not appear ($Nd_i, t_j = 0$), we set *w (i,j)=0*.
Once a weighting scheme has been selected, automated indexing can be performed by simply selecting the words that satisfy the given weight constraints for each document. [1]

2.2 The Association Rule Mining phase

2.2.1 Direct Hashing and Pruning

We used an algorithm for finding frequent itemsets in document databases. The basic idea of our algorithm is the Direct Hashing and Pruning (DHP) algorithm, which is in fact a variation of the well-known Apriori algorithm. In the DHP algorithm, a hash table is used in order to reduce the size of the candidate k+1 itemsets generated at each step.
Fig 2 shows the pseudocode of the DHP algorithm. [4]

Input: Database
Output: Frequent k-itemset

/* *Database = set of transactions;*
Items = set of items;
transaction = <TID, {x ∈ Items}>;
F_1 is a set of frequent l-itemsets */

$F_1 = \phi$;
/* *H_2 is the hash table for 2-itemsets*
Read the transactions, and count the occurrences of each item, and generate H_2 */

for each transaction $t \in$ Database do begin
  for each item x in t do
    $x.count++$;
  for each 2-itemset y in t do
    $H_2.add(y)$;
end
//*Form the set of frequent 1-itemsets*

for each item $i \in$ Items do
  if $i.count / |Database| \geq$ min sup
  then $F_1 = F_1 \cup i$;
end

/**Remove the hash values without the minimum support* */

$H_2.prune(\text{min sup})$;
/**Find F_k, the set of frequent k-itemsets, where $k \geq 2$* */

for each $(k := 2; F_{k-1} \neq \phi; k++)$ do begin

  // *C_k is the set of candidate k-itemsets*
  $C_k = \phi$;
/* *$F_{k-1} * F_{k-1}$ is a natural join of F_{k-1} and F_{k-1} on the first k - 2 items*
H_k is the hash table for k-itemsets */

  for each $x \in \{F_{k-1} * F_{k-1}\}$ do
    if $H_k.hassupport(x)$
    then $C_k = C_k \cup x$;
  end
  /**Scan the transactions to count candidate k-itemsets and generate H_{k+1}* */

  for each transaction $t \in$ Database do begin
    for each k-itemset x in t do
      if $x \in C_k$
      then $x.count++$;
    for each $(k+1)$-itemset y in t do
      if $\neg\exists z \mid z = k - \text{subset of } y \wedge \neg H_k.hassupport(z)$
      then $H_{k+1}.add(y)$;
  end
  // *F_k is the set of frequent k-itemsets*

  $F_k = \phi$;
  for each $x \in C_k$ do
    if $x.count / |Database| \geq$ min sup
    then $F_k = F_k \cup x$;
  end

  /* *Remove the hash values without the minimum support from H_{k+1}* */

  $H_{k+1}.prune(\text{min sup})$;
end
Answer = $\cup_k F_k$;

Fig 2. The Direct Hashing and Pruning Algorithm

2.3 The Document Clustering phase

Here, we have followed an effective approach for clustering a text corpus with the aid of association rules between the large frequent term sets. The proposed approach consists of the following major two steps: 1) Based on the strong association rules, the text documents are partitioned, and 2) Clustering of text documents from the partitions by means of representative words.

2.3.1 Partitioning the Text Documents based on Association Rules and Clustering:

All strong association rules generated from the DHP algorithm are used as input to this step. Strong association rules mean that all association rules that satisfies the confidence threshold.
This step consists of 4 main steps [2]:

2.3.1.1 Picking out all strong Association Rules

Initially, we sort the set of generated association rules in descending order in accordance with their confidence level.

2.3.1.2 Constructing Initial Partitions

An initial partition P1 is constructed for first association rule in Rs (sorted rules). Afterward, all the documents containing the two sides that constructed the rules are included in the same cluster. Next, we take the second association rules whose confidence is less than the previous one to form a new partition P2. This partition is formed by the same way of the partition P1. This procedure is repeated until every association rules moved into partition P_i since
$P_i = < R_i , doc [R_i] >$
Since a document usually contains more than one frequent termset, the same document may appear in multiple initial partitions, i.e., initial partitions are overlapping. The purpose of initial partitions is to ensure the property that all the documents in a cluster contain all the terms in the association rules that define the partition. These rules can be considered as the mandatory identifiers for every document in the partition. We use these association rules as the partition label to identify the partition .

2.3.1.3 Merging Similar Partitions

In this step, all partitions that contain the similar documents are merged into one partition. In other words; two identical partitions are merged to one. The benefit of this step is reducing the number of resulted partitions that are considered redundant.

2.3.1.4 Clustering

In this step, we remove the overlapping of partitions since there are some documents belong to one or more initial partitions. we assign a document to the "Optimal" initial partition so that each document belongs to exactly one partition. This step also guarantees that every document in the partition still contains the mandatory identifiers.
We propose the Weighted Score ($P_i \leftarrow doc_j$) in equation:

$$(P_i \leftarrow doc_j) = \sum_k w_k * m_i / n_w$$

to measure the optimal initial partition Pi for a document doc_j.

where $\sum_k wk$ represents the sum of weighted values of all words constructed the association rules from doc_j , m_i represents the number of documents in the initial partition P_i , and n_w represents the number of words that construct the partition p_i from doc_j. The weighted values of words w_k are defined by the standard inverse document frequency (TF-IDF) in the indexing process in section (2.1.3). The Weighted Score measure used the weighted values of frequent termsets instead of the number of occurrences of the terms in a document. Since the weighted values are an important piece of information based on the intuitive presumption of the weighting schema that is: the more often a term occurs in a document, the more representative of the content of the document (term frequency). Moreover the more documents the term occurs in, the less discriminating it is (inverse document frequency). To make partitions non-overlapping, we assign each doc_j to the initial partition P_i of the highest $score_i$. After this assignment, if there are more than one P_i that maximizes the Weighted Score **($P_i \leftarrow doc_j$)** , we will choose the one that has the most number of words in the partition label. After this step, each document belongs to exactly one partition that will be known as a Cluster.

4. Experimental Result

3.1 Offline Collecting of Documents

The first step we did is collecting and analyzing the documents (i.e. the relevant documents). The process of selecting documents in the is done offline that means the documents are previously downloaded. The largest Reuters datasets is an example for offline documents. The Reuters-21578 collection was our resource of datasets. We chose 14 text documents from 4 different categories with reasonable sizes to apply our test on.
The code may take a few minutes to proceed.

```
partition0
energy_0007434.txt
potato_0002178.txt
rice_0003164.txt
rice_0008101.txt
processing Done
```

Primary output of clustering for small no. of documents that generate only one strong association rule, more data will be present later.

3.2 Post Processing (Future Work)

In this section we're trying to get a suitable evaluation method for our clustering.
The F-measure, as the commonly used external measurement, is used to evaluate the accuracy of our clustering algorithms. F-measure is an aggregation of Precision and Recall concept of information retrieval. Recall is the ratio of the number of relevant documents retrieved for a query to the total number of relevant documents in the entire collection as:

$$Recall\left(K_i, C_j\right) = \frac{n_{ij}}{|K_i|}$$

Precision is the ratio of the number of relevant documents to the total number of documents retrieved for a query as:

$$Precision\left(K_i, C_j\right) = \frac{n_{ij}}{|C_j|}$$

while F-measure for cluster C_j and class K_i is calculated as in:

$$F\left(K_i, C_j\right) = \frac{2 * Recall\left(K_i, C_j\right) * Precision\left(K_i, C_j\right)}{Recall\left(K_i, C_j\right) + Precision\left(K_i, C_j\right)}$$

where n_{ij} is the number of members of class K_i in cluster C_i.$|C_j|$ is the number of members of cluster C_j and $|K_i|$is the number of members of class K_i .

The weighted sum of all maximum F-measures for all natural classes is used to measure the quality of a clustering result C. This measure is called the overall F-measure of C, denoted F(C) is calculated as in:

$$F(C) = \sum_{K_i \in K} \frac{|K_i|}{|D|} max_{C_j \in C}\{F(K_i, C_j)\}$$

where K denotes all natural classes; C denotes all clusters at all levels; $|K_i|$denotes the number of documents in natural class K_i ; and |D| denotes the total number of documents in the dataset. The range of F(C) is [0,1]. A large F(C) value indicates a higher accuracy of clustering.

5. Source code description

Attached with this report a source code written in java; that consists of 4 packages and 9 classes as follows:

4.1 mining.preprocessing

4.1.1 stop.java

4.1.2 stemmer.java

4.1.3 indexing.java

4.2 mining.directHashing

4.2.1 dhp.java

4.3 mining.partitioningNclustering

4.3.1 Rule.java

4.3.2 Rules.java

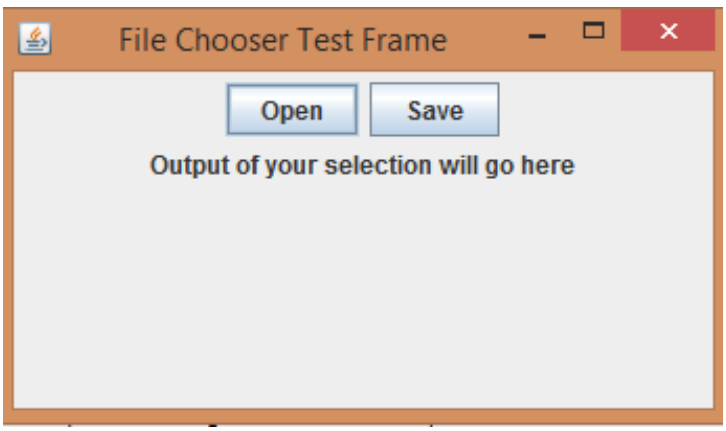
4.3.3 partitioning.java

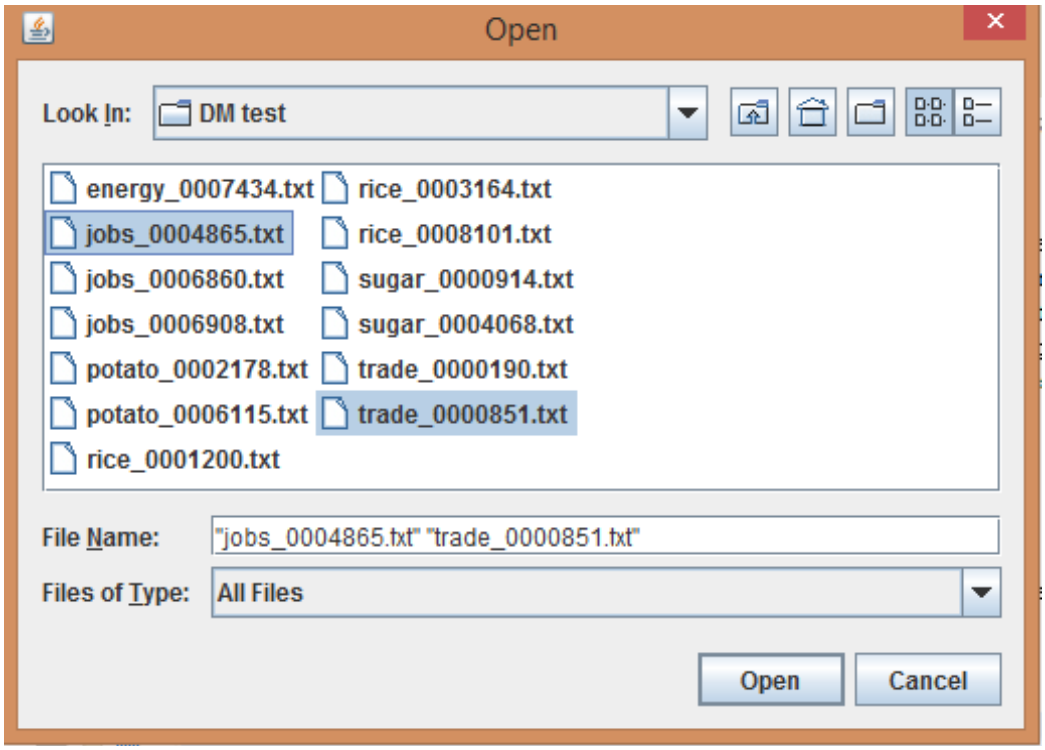
4.3.4 clustering.java

4.4 mining.testing

4.4.1 Main.java

In the main class we used JfileChooser as a GUI to choose our documents. As shown in the following figures:





This class is used to call all other classes' functions through created objects and test them according to the user input (weighting schema, minimum confidence and minimum support).

6. References

[1] N. Negm, P. Elkafrawy, A. Salem,An Efficient Hash-based Association Rule Mining Approach for Document Clustering.

[2] N. Negm, P. Elkafrawy, M.Amin, A. Salem, Investigate the Performance of Document Clustering Approach Based on Association Rules Mining, in IJACSA, Vol. 4, No. 8, 2013.

[3] J. S. Park, M. S. Chen and P. S. Yu, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules", IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 5, (Sep./Oct. 1997).

[4] S. Ozel, and H. Guvenir,An Algorithm for Mining Association Rules Using Perfect Hashing and Database Pruning, in Bilkent University, Department of Computer Engineering, Ankara, Turkey.
[5] S. Krishna, and S. Bhavani, An efficient approach for text clustering based on frequent itemsets, in Proc. of European J. of Scientific Research, vol.42, no.3, pp.399-410, 2010.
[6] K. Rajeswari, V. Vaithiyanathan, S. Tonge, R. Phalnikar, Mining Association Rules using Hash Table, International J. of Computer Applications (0975 – 8887), Vol 57– No.8, Nov 2012.