FUZZY CLUSTERING OF TEXT DOCUMENT USING KEYWORD EXTRACTION

JINTO JACOB, SIVADAS T NAIR

Viswajyothi college of Engineering, Kerala, India.

Abstract- Sentence clustering is the process of making similar sentences into a group. Clustering techniques is essential in the data mining process to reveal natural structures and identify interesting patterns in the underlying data. Cluster analysis seeks to partition a given data set into groups based on specified features so that the data points within a group are more similar to each other than the points in different groups. Fuzzy clustering algorithms allow patterns to belong to all clusters with differing degrees of membership. This is important in domains such as sentence clustering, since a sentence is likely to be related to more than one theme or topic present within a document or set of documents. The keyword extraction is the process of identifying important words in the documents. When applied to clustering, this will improve the performance. Our paper presents a new keyword extraction method and a framework for text clustering. This key word extraction method uses both semantic similarity as well as frequency of word occurrence. The method calculates two keyword extraction scores as a TFIDF score and semantic score. Both scores are then combined to form a new score and top keywords are extracted. The irrelevant words are removed from the sentences and similarity is calculated. The similarity is then passed to a clustering algorithm based on EM algorithm, which will return membership values of the sentences in each cluster. These values are then used to cluster sentences. The method was evaluated with different datasets and the result shows that our approach outperform the existing systems.

Keywords- Text Mining, Fuzzy Clustering, Text Clustering, Keyword Extraction.

I. INTRODUCTION

This world is full of data. Every day, people encounter a large amount of information and store or represent it as data, for further analysis and management. One of the vital means in dealing with these data is to classify or group them into a set of categories or clusters. Clustering algorithms partition data into a certain number of clusters (groups, subsets, or categories). Most researchers describe a cluster by considering the internal homogeneity and the external separation i.e., patterns in the same cluster should be similar to each other, while patterns in different clusters should not be similar. Sentence clustering plays an important role in many text processing activities. For example, incorporating sentence clustering into multidocument summarization helps avoid problems of content overlap which leads to better coverage. However, sentence clustering can also be used for text mining tasks. For example, in web mining the objective is to discover some information from a set of documents. Clustering the sentences of those documents would intuitively expect at least one of the clusters to be closely related to the concepts described by the query terms; however, other clusters may contain information pertaining to the query in some way unknown. If the information in such data are found it would successfully mined new information. Irrespective of the specific task most documents will contain interrelated topics or themes, and many sentences will be related to some degree to a number of these. The successful capture of such fuzzy relationships will lead to an increase in the breadth and scope of problems to which sentence clustering can be applied. However, clustering text at the sentence level poses specific challenges not present when clustering larger segments of text, such as documents. Clustering text at the document level is well established in the Information Retrieval (IR), where documents are typically represented as data points in a high dimensional vector space in which each dimension corresponds to a unique keyword, leading to a rectangular representation in which rows represent documents and columns represent attributes of those documents (e.g., tf-idf values of the keywords). This type of data is amenable to clustering by a large range of algorithms. Since data points lie in a metric space, prototype-based algorithms can be applied, which represent clusters in terms of parameters such as means and covariance, and therefore assume a common metric input.

II. PROPOSED SYSTEM MODEL

The proposed system is a new method for keyword extraction and fuzzy clustering. This entire system is divided into four parts as: keyword extraction, similarity calculation, membership calculation and clustering.

The keyword extraction method is described in section 3. The paper then describe about the similarity calculation method which uses a semantic similarity calculation in section 4. Section 5 of paper presents the clustering algorithm which uses EM-based algorithm for clustering the text document. The final section discusses the issues with the threshold selection and various implementation issues. A detailed evaluation of the method is also given in the paper.
III. KEYWORD EXTRACTION

This world is full of data. Every day, people encounter a large amount of information and store or represent it as data, for further analysis and

![Fig.1. Keyword Extraction](image)

A. TF-IDF Calculation

In a document the frequency based methods calculate scores based on number of occurrence of the word in the document or sentence. The frequency based score uses a ‘m×n’ matrix for calculation of score, where ‘m’ is the number of sentences in the document, ‘n’ is the number of terms and each entry in the matrix is the weight of the jth term in the ith sentence. The weight of a term in a sentence is calculated as the product of local and global weights. The local weighting function is used to measures the importance of a term within a sentence and the global weighting function is used to measures the importance of a term in the entire document.

The term frequency (TF) is the total number of occurrence of the word in a sentence. The IDF is the inverse document frequency. The IDF will be less for the frequent and unimportant words.

B. Semantic Score Calculation

Submit In the semantic score the semantic similarity is used for finding importance of a word. Lots of semantic similarity measures are developed in the recent years. These methods can be classified into two groups as: edge count based methods and information theory based methods. The semantic similarity methods calculate the similarity using a hierarchical structure which represents the entire knowledge base as a tree structure. Some semantic knowledge bases readily available are WordNet, Spatial Date Transfer Standard, and Gene Ontology. The method in the paper uses the WordNet for the similarity calculation. The WordNet arrange words in a tree or a graph format. The similarity used here is the Resnik’s similarity.

C. Combining Scores

Submit the scores can now be combined to get a new and effective score. The combined score can be calculated as:

\[
\text{Score}_i = \alpha \cdot \text{SemScore}_i + (1 - \alpha) \cdot \text{TFIDF}_i \tag{1}
\]

Where SemScore is the semantic score of word i and \( \alpha \) is the importance factor. When \( \alpha = 0.5 \) we get both measure equal priority. As \( \alpha \) increase semantic score will get more priority. The words are now sorted according to the score and the words with high scores are returned as keywords.

IV. SIMILARITY CALCULATION

After the keyword extraction all the non-keywords are removed from the document and the sentences are represented only with keywords. The similarity calculation step calculates the sentence to sentence similarity using the document thus created. Unlike classical methods that use a precompiled word list, this method dynamically forms the semantic vectors solely based on the compared sentences. The similarity calculation contains three steps as: Calculating Lexical Vector, calculating semantic vector and calculating the similarity. They can be explained as below.

A. Lexical Vector

The lexical vector can be calculated as first step to find similarity. Given two sentences, T1 and T2, a joint word set is formed: T = T1 U T2 = \{w1, w2,...,wm\}. The joint word set T contains all the distinct words from T1 and T2. Since a word may appear in a sentence with different forms that convey a specific meaning, the words are represented in the form they appear in the sentence. For example, dog and dogs, man and men are considered as four distinct words and are included in the joint word set. Since the joint word set is purely derived from the compared sentences, it is contain no redundant information. Each sentence is represented by the use of the joint word set. The vector derived from the joint word set is called the lexical vector, denoted by ‘s’. The dimension of the vector is equal to the number of words in the word set and each entry corresponds to a word in the joint word set. The value of an entry of the lexical vector is determined by the semantic similarity of the word to the words in the sentence. i.e.,

Case 1. If word wi is in the sentence, si is set to 1.
Case 2. If wi is not contained in the sentence, a semantic similarity score is computed between wi and each word in the sentence T1. The most similar word in T1 to wi is find out and its value is inserted as entry to si

B. Semantic Vector

Submit The function words in the joint word set contribute less to the meaning of a sentence than other words. Thus, a scheme is needed to weight each word. The information content is used to find significance of each word. Thus frequent words have low value of information content than low frequent items. The information content of a word is derived from its
probability in a corpus. Each cell is weighted by the associated information I(wi) and I(~ wi). Finally, the value of an entry of the semantic vector is:

\[ \text{sem}_i = s \cdot i(w_i) \cdot i(~w_i) \] (2)

Table-1
F-Measure

<table>
<thead>
<tr>
<th>Document</th>
<th>FRECCA</th>
<th>New Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.4105</td>
<td>0.55</td>
</tr>
<tr>
<td>S2</td>
<td>0.3705</td>
<td>0.5010</td>
</tr>
<tr>
<td>S3</td>
<td>0.3665</td>
<td>0.4387</td>
</tr>
<tr>
<td>S4</td>
<td>0.4723</td>
<td>0.5055</td>
</tr>
</tbody>
</table>

where wi is a word in the joint word set, ~ wi is its associated word in the sentence. The use of I(wi) and I(~ wi) allows the concerned two words to contribute to the similarity based on their individual information contents.

C. Calculating Similarity

The semantic similarity between two sentences is defined as the cosine coefficient between the two vectors. This similarity will be in the form of a n * n matrix, where n is number of sentences. This is now given to a fuzzy clustering algorithm.

V. FUZZY CLUSTERING

Contents

The fuzzy clustering algorithm used is the FRECCA algorithm. FRECCA uses the PageRank score of an object within a cluster as a measure of its centrality to that cluster. The main PageRank methods for sentences are TexRank and LexRank. These PageRank values are then treated as likelihoods. Since there is no parameterized likelihood function as such, the only parameters that need to be determined are the cluster membership values and mixing coefficients. The algorithm uses Expectation Maximization to optimize these parameters. Assume in the following that the similarities between objects are stored in a similarity matrix S = [sij], where sij is the similarity between objects i and j. This is done in 3 steps as Initialization, Expectation and Maximization. The steps are explained in following sections

A. Initialization

Assume here that cluster membership values are initialized randomly, and normalized such that cluster membership for an object sums to unity over all clusters. Mixing coefficients are initialized such that priors for all clusters are equal. An alternative to random initialization is to initialize cluster membership values with values found by first applying a computationally inexpensive hard clustering algorithm such as Spectral Clustering or k-Medoids. This will result in each object having an initial membership value of either 0 or 1 to each cluster.

B. Expectation

The E-step (Expectation step) calculates the PageRank value for each object in each cluster. PageRank values for each cluster are calculated with the affinity matrix weights Wij obtained by scaling the similarities by their cluster membership values; i.e.,

\[ W_{ij}^m = s_{ij} \cdot p_i^m \cdot p_j^m \] (3)

where \( W_{ij}^m \) is the weight between objects i and j in cluster m, sij is the similarity between objects i and j, and pm and pmj are the respective membership values of objects i and j to cluster m. Using this value calculate the PageRank. Once PageRank scores have been determined, these are used to calculate cluster membership values. The expectation step will be done until a convergence is achieved.

C. Maximization

The maximization step involves only the single step of updating the mixing coefficients based on membership values calculated in the Expectation Step.

VI. CLUSTERING

The output of the algorithm is in the form of a matrix of size s*m where s is number of the sentence and m is number of clusters. Each entry will be the membership value of the sentence in the cluster. There are two types of clustering possible using this matrix, a fuzzy clustering and hard clustering. In fuzzy clustering the sentences are assigned to all clusters in which its membership value is above a threshold allowing some sentence to be in multiple clusters. In hard clustering the sentence is assigned to cluster for which membership value is highest. This algorithm is able to find out overlapping clusters in semantic sentences. Potential application of the algorithm is document summarization and text mining. The main problem is that the performance is depending on the quality of inputs. The method is not sensitive to the initialization of cluster membership values and it appears to be able to converge to an appropriate number of clusters, even if the number of initial clusters was set very high. The algorithm can also be applied to asymmetric matrix.
VII. EXPERIMENTAL EVALUATION

Strongly, the proposed fuzzy clustering and keyword extraction mechanism has been evaluated and compared with an existing method. The objective is to evaluate the performance of the proposed technique with respect to a database of different documents. All the experiments were conducted on an Intel I3 2.40 GHz PC with 2 GB of memory. All the algorithms were implemented using JAVA.

A. Analysis of Performance

The method is compared with FRECCA algorithm to compare performance. The main comparison was in terms of accuracy, Precision, recall and F measure. The algorithm was tested with many documents in which four comparisons are given here. The results show that the proposed system is better. Precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance. In simple terms, high precision means that an algorithm returned substantially more relevant results than irrelevant, while high recall means that an algorithm returned most of the relevant results.

B. Analysis of importance factor

The importance factor is used to give importance to the similarity measure in keyword extraction, as the value increases the semantic similarity will get more importance.

The value here is obtained by setting different parameters as: \( d=0.85 \), \( n=75 \%

C. Analysis of Keyword Threshold

The keyword threshold is used to determine how many keywords should be extracted. If the value is too low the words taken will be low and information may lose. Different values are taken and evaluation is given.

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Importance factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>Accuracy</td>
</tr>
<tr>
<td>0.3</td>
<td>0.5510</td>
</tr>
<tr>
<td>0.4</td>
<td>0.5510</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6326</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6122</td>
</tr>
<tr>
<td>0.7</td>
<td>0.6122</td>
</tr>
</tbody>
</table>

CONCLUSION

As an important tool for data exploration, cluster analysis examines unlabeled data, by either constructing a hierarchical structure, or forming a set of groups according to a pre-specified number. Usually, algorithms are designed with certain assumptions and favor some type of biases. In this sense, it is not accurate to say best in the context of clustering algorithms, although some comparisons are possible. The work proposes a method which is able to find out overlapping clusters in sentences. Potential application of the algorithm is document summarization and text mining. The keyword extraction used in the system will remove the unimportant words in the sentence and will improve the performance. Unlike old methods the new keyword extraction step uses both frequency and semantic similarity for keyword extraction. The algorithm appears to be able to converge to an appropriate number of clusters. The algorithm can also be applied to symmetric and asymmetric matrix. The experimental evaluation shows that the method works better than previous methods.

REFERENCES


Fuzzy Clustering of Text Document Using Keyword Extraction


