Text Documents clustering using K Means Algorithm

Mrs Sanjivani Tushar Deokar
Assistant professor
sanjivanideokar@gmail.com

Abstract: With the advancement of technology and reduced storage costs, individuals and organizations are tending towards the usage of electronic media for storing textual information and documents. It is time consuming for readers to retrieve relevant information from unstructured document collection. It is easier and less time consuming to find documents from a large collection when the collection is ordered or classified by group or category. The problem of finding best such grouping is still there. This paper discusses the implementation of K-Means clustering algorithm for clustering unstructured text documents that we implemented, beginning with the representation of unstructured text and reaching the resulting set of clusters. Based on the analysis of resulting clusters for a sample set of documents, we have also proposed a technique to represent documents that can further improve the clustering result.

Index terms: K Means, document Vector, Residual sum of square, Tf-IDF.

1. Introduction
Clustering can be considered the most important unsupervised learning problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”.
A cluster is therefore a collection of objects which are coherent internally, but clearly dissimilar to the objects belonging to other clusters.

2. Classification
Clustering algorithms may be classified as listed below:

Flat clustering: Creates a set of clusters without any explicit structure that would relate clusters to each other; It’s also called exclusive clustering Hierarchical clustering: Creates a hierarchy of clusters.
Hard clustering: Assigns each document/object as a member of exactly one cluster.
Soft clustering: Distribute the document/object over all users.

Algorithms
Agglomerative (Hierarchical clustering)
K-Means (Flat clustering, Hard clustering)
EM Algorithm (Flat clustering, Soft clustering)
Hierarchical Agglomerative Clustering (HAC) and K-Means algorithm have been applied to text clustering in a straightforward way. Typically it usages normalized, TF-IDF-weighted vectors and cosine similarity. Here, the k-means algorithm using a set of points in n-dimensional vector space for text clustering

3. K-Means Algorithm

The k-means clustering algorithm is known to be efficient in clustering large data sets. This clustering algorithm was developed by MacQueen, and is one of the simplest and the best known unsupervised learning algorithms that solve the well-known clustering problem. The K-Means algorithm aims to partition a set of objects, based on their attributes/features, into k clusters, where k is a predefined or user-defined constant. The main idea is to define k centroids, one for each cluster. The centroid of a cluster is formed in such a way that it is closely related (in terms of similarity function; similarity can be measured by using different methods such as cosine similarity, Euclidean distance, Extended Jaccard) to all objects in that cluster.
3.1 Basic K-Means Algorithm
Choose k number of clusters to be Determined.
Choose k objects randomly as the initial cluster center
Repeat
3.1. Assign each object to their closest cluster
3.2. Compute new clusters, i.e.
   Calculate mean points.
4. Until
   4.1. No changes on cluster centers
      (i.e. Centroids do not change location any more) OR
   4.2. No object changes its cluster

The document is represented in the form of vector such that the words (also called features) represent dimensions of the vector and frequency of the word in document is the magnitude of the vector i.e.
Vector is of the form
\[ \langle t_1, f_1 \rangle, \langle t_2, f_2 \rangle, \langle t_3, f_3 \rangle, \ldots, \langle t_n, f_n \rangle \]
Where \( t_1, t_2, \ldots, t_n \) are the terms/words (dimension of the vector) and \( f_1, f_2, \ldots, f_n \) are the corresponding frequencies or magnitude of the vector components.
The algorithm of creating a document vector is given below:
Input
Token Stream (TS), all the tokens in the document Collection
Output
HS, a Hash Table of tokens with respective frequencies
Initialize:
Hash Table (HS):= empty Hash Table
for each Token in Token Stream (TS) do
  If Hash Table (HS) contains Token then
    Frequency:= value of Token in hs
    increment Frequency by 1
  else
    Frequency:=1
  end if
end for
Store Frequency as value of Token in Hash Table (HS)

3.2 Residual Sum of Squares
RSS is the objective function in K-means and our goal is to minimize it. Because N is fixed, minimizing RSS is equivalent to minimizing the average squared distance; a measure of how well centroids represent their documents. RSS is a measure of how well the centroids represent the members of their clusters, the squared distance of each vector from its centroid summed over all vectors.

\[
RSS_k = \sum_{\omega_k} | \vec{x} - \bar{\mu}(\omega_k) |^2
\]

Where
- \( \omega_k \) Document cluster k
- \( \bar{\mu} \) Mean or centroid of the documents in cluster \( \omega_k \)
- \( \vec{x} \) Document vector in cluster k

The algorithm then moves the cluster centers around in space in order to minimize RSS.

The first step of K-means is to select as initial cluster centers K randomly selected documents, the seeds. The algorithm then moves the cluster centers seed around in space to minimize RSS. This is done iteratively by repeating two steps until a stopping criterion is met: Reassigning documents to the cluster with the closest centroid and recomputing each centroid based on the current members of its cluster.
Assignment of documents to clusters (the partitioning function) does not change between iterations. Except for cases with a bad local minimum, this produces a good clustering, but run-time may be unacceptably long. Terminate when the decrease in RSS falls below a threshold \( R \). For small \( R \), this indicates that we are close to convergence. Again, we need to combine it with a bound on the number of iterations to prevent very long run-times. We now show that K-means converges by proving that RSS monotonically decreases in each iteration. We will use decrease in the meaning decrease or does not change in this section. First, RSS decreases in the reassignment step; each vector is assigned to the closest centroid, so the distance it contributes to RSS decreases. Second, it decreases in the re-computation step because the new centroid is the vector \( v \) for which RSS\(_k\) reaches its minimum.

where \( x_m \) and \( v_m \) are the mth components of their respective vectors. Setting the partial derivative to zero, we get:

\[
v_m = \frac{1}{|\omega_k|} \sum_{\mathbf{x} \in \omega_k} x_m
\]

which is the component wise definition of the centroid. Thus, we minimize RSS\(_k\) when the old centroid is replaced with the new centroid. RSS, the sum of the RSS\(_k\), must then also decrease during recomputation. Because there is only a finite set of possible clustering’s, a monotonically decreasing algorithm will eventually arrive at a (local) minimum. Take care, however, to break ties consistently, for example, by assigning a document to the cluster with the lowest index if there are several equidistant centroids. Otherwise, the algorithm can cycle forever in a loop of clustering’s that have the same cost. Although this proves the convergence of K-means, there is unfortunately no guarantee that a global minimum in the objective function will be reached.

### 3.3 Termination Condition

We can apply one of the following termination conditions. A fixed number of iterations \( I \) has been completed. This condition limits the runtime of the clustering algorithm, but in some cases the quality of the clustering will be poor because of an insufficient number of iterations. Assignment of documents to clusters does not change between iterations. Except for cases with a bad local minimum, this produces a good clustering, but runtimes may be unacceptably long. Centroids do not change between iterations. This is equivalent to not terminate when RSS falls below a threshold. This criterion ensures that the clustering is of a desired quality after termination. In practice, we need to combine it with a bound on the number of iterations to guarantee termination. Terminate when the decrease in RSS falls below a threshold \( t \). For small \( t \), this indicates that we are close to convergence. Again, we need to combine it with a bound on the number of iterations to prevent very long runtimes.

### 3.4 Bad choice of initial seed

In K-Means algorithm there is unfortunately no guarantee that a *global minimum* in the objective function will be reached, this is a particular problem if a document set contains many outliers, documents that are far from any other documents and therefore do not fit well into any cluster. Frequently, if an outlier is chosen as an initial seed, then no other vector is subsequent iterations. Thus, we end up with a singleton cluster (a cluster with only one document) even though there is probably a clustering with lower RSS.

*Effective heuristics for seed selection include:*

Excluding outliers from the seed set Trying out multiple starting points and choosing
the clustering with the lowest cost; and Obtaining seeds from another method such as hierarchical clustering.

3.5 TF-IDF

TF-IDF stands for term frequency-inverse document frequency, is a numerical statistics which reflects how important a word is to a document in a collection or corpus, it is the most common weighting method used to describe documents in the Vector Space Model, particularly on IR problems.

The number of times a term occurs document is called its term frequency. We can calculate the term frequency for a word. The inverse document frequency is a measure of whether the term is common or rare across all documents. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

as the ratio of number of times the word occurs in the document to the total number of words in the document.

Multiplying together these two metrics gives the tf-idf, placing importance on terms frequent in the document and rare in the corpus.

The tf*idf of term t in document d is calculated as:

$$tf_{t,d} \times idf_t$$

e.g. For three vectors (after removing stop-words and performing stemming).
Doc1 < (computer, 60), (JAVA, 30)> Doc2 < (computer, 55), (PASCAL, 20)> Doc3 < (graphic, 24), (Database, 99)>

Total Documents, \(N=3\)

The vectors shown above indicate that the term 'computer' is less important compared to other terms (such as 'JAVA' which appears in only one document out of three) for identifying groups or clusters because this term appears in more number of documents (two out of three in this case) making it less distinguishable feature for clustering. Whatever the actual frequency of the term may be, some weight must be assigned to each term depending on the importance in the given set of documents. The method used in our implementation is the tf-idf formulation.

In tf-idf formulation the frequency of term \(i\), \(tf(i)\) is multiplied by a factor calculated using inverse-document - frequency \(idf (i)\) given in (2). In the example above, total number of documents is \(N=3\), the term frequency of 'computer' is \(tf_{computer}\) and the number of documents in which the term 'comCLUSTERING RESULT

An input sample of 24 documents [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34] were provided to the K-Means Algorithm. With the initial value of \(k=24\), the algorithm was run for three different scenarios:

(a) When the document vectors were formed on the basis of features (words) of the document.
(b) When the document vectors were formed on the basis of sub-category of features.
(c) When the document vectors were formed on the basis of parent category of the feature.

The result of k-means clustering algorithm for each case is given below:

<table>
<thead>
<tr>
<th>Cluster name</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Mining</td>
<td>[12, 13, 14, 16, 17, 18, 28]</td>
</tr>
<tr>
<td>Databases</td>
<td>[11, 25, 26, 27]</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>[23, 32]</td>
</tr>
<tr>
<td>Mobile Computing</td>
<td>[22, 24]</td>
</tr>
<tr>
<td>Microprocessors</td>
<td>[33, 34]</td>
</tr>
<tr>
<td>Programming</td>
<td>[30, 31]</td>
</tr>
<tr>
<td>Data Structures</td>
<td>[29]</td>
</tr>
<tr>
<td>Business Computing</td>
<td>[20, 21]</td>
</tr>
<tr>
<td>World Wide Web</td>
<td>[15]</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>[19]</td>
</tr>
</tbody>
</table>

computer' occurs is dfcomputer
4. CONCLUSION
In this paper I have discussed the concept of document clustering. I have also presented the implementation technique of K-means clustering algorithm. I have compared three different ways of representing a document and suggested how an organized domain dictionary can be used to achieve better similarity results of the documents. The implementation discussed in this paper considered residual sum of square and TF-IDF methods. This could further improve similarity measure of documents which would ultimately provide better clusters for a given set of documents.

5. References


