Analysis of Similarity Measures for Text Clustering

N. Sandhya
Associate Professor,CSE Dept
Gokaraju Rangaraju Institute of Engineering &Technology nadella_sandhya@yahoo.co.in
Hyderabad, 500072,India

Y.Sri Lalitha
Associate Professor,CSE Dept
Gokaraju Rangaraju Institute of Engineering &Technology lalitha_ysl@yahoo.co.in
Hyderabad, 500072,India

Dr.A.Govardhan
Principal
JNTUH College of Engineering govardhan_cse@yahoo.co.in
Jagtial,505501,India

Dr.K.Anuradha
Professor& Head,CSE Dept
Gokaraju Rangaraju Institute of Engineering &Technology kodali.anuradha@yahoo.com
Hyderabad, 500072,India

Abstract

Text document clustering plays an important role in providing intuitive navigation and browsing mechanisms by organizing large amounts of information into a small number of meaningful clusters. Clustering method has to embed the documents in a suitable similarity space. While several clustering methods and the associated similarity measures have been proposed in the past, there is no systematic comparative study of the impact of similarity measures on cluster quality. This may be because the popular cost criteria do not readily translate across qualitatively different measures. In this paper we compare four popular similarity measures (Euclidean, cosine, Pearson correlation and extended Jaccard) in conjunction with different types of vector space representation (boolean, term frequency and term frequency and inverse document frequency) of documents. Clustering of documents is performed using generalized k-Means; a Partitioned based clustering technique on high dimensional sparse data representing text documents.

Performance is measured against a human-imposed classification of Topic and Place categories. We conducted a number of experiments and used entropy measure to assure statistical significance of results. Cosine, Pearson correlation and extended Jaccard similarities emerge as the best measures to capture human categorization behavior, while Euclidean measures perform poor.

Keywords: Text clustering, Similarity Measures, Cluster Accuracy

---

1. INTRODUCTION

With ever increasing volume of text documents, the abundant texts flowing over the Internet, huge collections of documents in digital libraries and repositories, and digitized personal information such as blog articles and emails are piling up quickly every day. For text documents, clustering has proven to be an effective approach and an interesting research probleml. Clustering of text documents plays a vital role in efficient Document Organization, Summarization, Topic Extraction and Information Retrieval. Initially used for improving the precision or recall in an Information Retrieval System [1,2], more recently, clustering has been proposed for use in browsing a collection of documents [3] or in organizing the results

returned by a search engine in response to user's query [4] or help users quickly identify and focus on the relevant set of results. Customer comments are clustered in many online stores, such as Amazon.com to provide collaborative recommendations. In collaborative bookmarking or tagging, clusters of users that share certain traits are identified by their annotations. Document clustering has also been used to automatically generate Hierarchical clusters of documents [5]. The automatic generation of taxonomy of Web documents as the one provided by Yahoo! (www.yahoo.com) is often cited as a goal.

This paper is organized as follows. The section 2 deals with the related work in text document clustering, section 3 describes the document representation used in the experiments. Section 4 discusses the similarity measures and their semantics. Section 5 presents the K-means clustering algorithm and Section 6 explains experiment settings, evaluation approaches, results and analysis and Section 7 concludes and discusses future work.

2. RELATED WORK

Many clustering techniques have been proposed in the literature. Clustering algorithms are mainly categorized into Hierarchical and Partitioning methods [2, 3, 4, 5]. Hierarchical clustering method works by grouping data objects into a tree of clusters [6]. These methods can further be classified into agglomerative and divisive Hierarchical clustering depending on whether the Hierarchical decomposition is formed in a bottom-up or top-down fashion.K-means and its variants [7, 8, 9] are the most well-known partitioning methods [10].

Hierarchical clustering is often portrayed as the better quality clustering approach, but is limited because of its quadratic time complexity. In contrast, K-means and its variants have a time complexity which is linear in the number of documents, but are thought to produce inferior clusters.

Hierarchical techniques produce a nested sequence of partitions, with a single, all inclusive cluster at the top and singleton clusters of individual points at the bottom. Each intermediate level can be viewed as combining two clusters from the next lower level (or splitting a cluster from the next higher level). The result of a Hierarchical clustering algorithm can be graphically displayed as tree, called a dendogram.

In contrast to Hierarchical techniques, Partitional clustering techniques create a one-level (un-nested) partitioning of the data points. If K is the desired number of clusters, then Partitional approaches typically find all K clusters at once. Contrast this with traditional Hierarchical schemes, which bisect a cluster to get two clusters or merge two clusters to get one. Of course, a Hierarchical approach can be used to generate a flat partition of K clusters, and likewise, the repeated application of a Partitional scheme can derive Hierarchical clustering.

There are a number of Partitional techniques, but we shall only describe the K-means algorithm which is widely used in document clustering. K-means is based on the idea that a center point can represent a cluster. In particular, for K-means we use the notion of a centroid, which is the mean or median point of a group of points. Note that a centroid almost never corresponds to an actual data point. The algorithm is discussed in detail in section 5

3. DOCUMENT REPRESENTATION

In order to reduce the complexity of the documents and make them easier to handle, the document have to be transformed from the full text version to a document vector which describes the contents of the document. The representation of a set of documents as vectors in a common vector space is known as the vector space model. In the vector space model of IR, documents are represented as vectors of features representing the terms that occur within the collection. It is also termed as bag of words, where words are assume to appear independently and the order is immaterial. The value of each feature is called the term weight and is usually a function of term's frequency (or tf-idf) in the document, along with other factors.

If we consider each document as a multi-dimensional vector and then try to cluster documents based on their word contents, the problem differs from classic clustering scenarios in several ways. Document

clustering data is high dimensional, characterized by a highly sparse word-document matrix with positive ordinal attribute values and a significant amount of outliers. Here we use the frequency of each term as its weight, which means terms that appear more frequently are more important and descriptive for the document.

Vector Space representation of a document involves three steps. First step is the document indexing where content bearing terms are extracted from the documents. The second step is to compute the weights of indexed terms to enhance retrieval of documents relevant to the user. The final step is identifying the similarities between the documents.

3.1 Extracting Index Terms

It involves preprocessing text documents, apply stemming, remove stop words and tokenize the text. Documents in vector space can be represented using Boolean, Term Frequency and Term Frequency – Inverse Document Frequency.

In Boolean representation, if a term exists in a document, then the corresponding term value is set to one otherwise it is set to zero. Boolean representation is used when every term has equal importance and is applied when the documents are of small size.

In Term Frequency and Term Frequency Inverse Document Frequency the term weights have to be set. The term weights are set as the simple frequency counts of the terms in the documents. This reflects the intuition that terms occur frequently within a document may reflect its meaning more strongly than terms that occur less frequently and should thus have higher weights.

Each document d is considered as a vector in the term-space and represented by the term frequency (TF) vector:

$$d_{tf} = [\, tf_1 , tf_2 , \ldots\ldots\ldots tf_D \,]$$

where tf_i is the frequency of term i in the document and D is the total number of unique terms in the text database.

The second factor is used to give a higher weight to words that only occur in a few documents. Terms that are limited to few documents are useful for discriminating those documents from the rest of the collection, while terms that occur frequently across the entire collection aren't helpful. The inverse document frequency term weight is one way of assigning higher weights to these more discriminative words. IDF is defined via the fraction N/n_i, where, N is the total number of documents in the collection and n_i is the number of documents in which term i occurs.

Due to the large number of documents in many collections, this measure is usually squashed with a log function. The resulting definition IDF is thus:

$$\mathrm{idf}_i = \log\left(\frac{N}{n_i}\right)$$

Combining term frequency with IDF results in a scheme known as tf-idf weighting.

$$w_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$$

Thus, the tf–idf representation of the document d is:

$$d_{tf-idf} = [\, tf_1 \log(n / df_1),\ tf_2 \log(n / df_2),\ldots..,\ tf_D \log(n / df_D)]$$

To account for the documents of different lengths, each document vector is normalized to a unit vector (i.e., $||d_{tf\text{-}idf}||=1$). In the rest of this paper, we assume that this vector space model is used to represent

documents during the clustering. Given a set C_j of documents and their corresponding vector representations, the centroid vector c_j is defined as:

$$c_j = \frac{1}{|C_j|} \sum_{d_i \in C_j} d_i$$

where each d_i is the document vector in the set C_j, and j is the number of documents in Cluster C_j. It should be noted that even though each document vector d_i is of unit length, the centroid vector c_j is not necessarily of unit length. In this paper we experimented with three vector model representations Boolean, Term Frequency and Inverse Document Frequency vector space model.

4. SIMILARITY MEASURES

Document clustering groups similar documents to form a coherent cluster. However, the definition of a pair of documents being similar or different is not always clear and normally varies with the actual problem setting. For example, when clustering research papers, two documents are regarded as similar if they share similar thematic topics. When clustering is employed on web sites, we are usually more interested in clustering the component pages according to the type of information that is presented in the page. For instance, when dealing with universities web sites, we may want to separate professor's home pages from student's home pages, and pages for courses from pages for research projects. This kind of clustering benefits further analysis and utilize the dataset such as information retrieval and information extraction, by grouping similar types of information sources together.

Accurate clustering requires a precise definition of the closeness between a pair of objects, in terms of either the pair wise similarity or distance. A variety of similarity or distance measures have been proposed and widely applied, such as cosine similarity, Jaccard coefficient, Euclidean distance and Pearson Correlation Coefficient.

4.1 Cosine Similarity Measure

For document clustering, there are different similarity measures available. The most commonly used is the cosine function. For two documents d_i and d_j, the similarity between them can be calculated

$$cos(d_i, d_j) = \frac{d_i \cdot d_j}{\| d_i \| \| d_j \|}$$

Since the document vectors are of unit length, the above equation is simplified to:

$$cos(d_i, d_j) = d_i \cdot d_j$$

When the cosine value is 1 the two documents are identical, and 0 if there is nothing in common between them (i.e., their document vectors are orthogonal to each other).

4.2 Jaccard Coefficient

The Jaccard coefficient, which is sometimes referred to as the Tanimoto coefficient, measures similarity as the intersection divided by the union of the objects. For text document, the Jaccard coefficient compares the sum weight of shared terms to the sum weight of terms that are present in either of the two documents but are not the shared terms.

The Cosine Similarity may be extended to yield Jaccard Coeff. in case of Binary attributes

$$\text{Jaccard Coff (A,B)} = \frac{\sum_i A_i \,.\, B_i}{\sum_i ||A_i||^2 + \sum_i ||B_i||^2 - \sum_i A_i * B_i}$$

$$\text{Jaccard Index (A, B)} = \frac{A \cap B}{AU\ B}$$

4.3 Euclidean Similarity

This is the most usual, "natural" and intuitive way of computing a distance between two samples. It takes into account the difference between two samples directly, based on the magnitude of changes in the sample levels. This distance type is usually used for data sets that are suitably normalized or without any special distribution problem.

$$\text{Euclidean Distance (A, B)} = \sqrt{\sum_i (A_i - B_i)^2}$$

$$\text{Euclidean Similarity (A, B)} = 1 - \sqrt{\sum_i (A_i - B_i)^2}$$

4.4 Pearson Correlation Coefficient

This distance is based on the Pearson correlation coefficient that is calculated from the sample values and their standard deviations. The correlation coefficient 'r' takes values from –1 (large, negative correlation) to +1 (large, positive correlation). Effectively, the Pearson distance -dp- is computed as dp = 1 - r and lies between 0 (when correlation coefficient is +1, i.e., the two samples are most similar) and 2 (when correlation coefficient is -1).

$$SIM_P(\overrightarrow{t_a}, \overrightarrow{t_b}) = \frac{m\sum_{t=1}^{m} w_{t,a} \times w_{t,b} - TF_a \times TF_b}{\sqrt{[m\sum_{t=1}^{m} w_{t,a}^2 - TF_a^2][m\sum_{t=1}^{m} w_{t,b}^2 - TF_b^2]}}$$

$$\text{where } TF_a = \sum_{t=1}^{m} w_{t,a} \text{ and } TF_b = \sum_{t=1}^{m} w_{t,b}.$$

Where t_a and t_b are m-dimensional vectors over the term set $T = \{t_1, \ldots, t_m\}$.

The Euclidean distance is a distance measure, while the cosine similarity, Jaccard coefficient and Pearson coefficient are similarity measures. We apply a simple transformation to convert the similarity measure to distance values. Because both cosine similarity and Jaccard coefficient are bounded in [0, 1] and monotonic, we take D = 1 – SIM as the corresponding distance value. For Pearson coefficient, which ranges from –1 to +1, we take D = 1 – SIM when SIM ≥ 0 and D = |SIM| when SIM < 0.

5. CLUSTERING ALGORITHM

For subsequent experiments, the standard K-means algorithm is chosen as the clustering algorithm. This is an iterative Partitional clustering process that aims to minimize the least squares error criterion [6]. As mentioned previously, Partitional clustering algorithms have been recognized to be better suited for handling large document datasets than Hierarchical ones, due to their relatively low computational requirements [7, 9, 8]. The standard K-means algorithm works as follows. Given a set of data objects D and a pre-specified number of clusters k, k data objects are randomly selected to initialize k clusters, each one being the centroid of a cluster. The remaining objects are then assigned to the cluster represented by the nearest or most similar centroid. Next, new centroids are recomputed for each cluster and in turn all documents are re-assigned based on the new centroids. This step iterates until a converged and fixed solution is reached, where all data objects remain in the same cluster after an update of centroids. The generated clustering solutions are locally optimal for the given data set and the

initial seeds. Different choices of initial seed sets can result in very different final partitions. Methods for finding good starting points have been proposed [10]. However, we will use the basic K-means algorithm because optimizing the clustering is not the main focus of this paper.

The K-means algorithm works with distance measures which basically aims to minimize the within-cluster distances. Therefore, similarity measures do not directly fit into the algorithm, because smaller values indicate dissimilarity.

1. Select K points as the initial centroids.

2. Assign all points to the closest centroid.

3. Recompute the centroid of each cluster.

4. Repeat steps 2 and 3 until the centroids don't change.

6. EXPERIMENT

It is very difficult to conduct a systematic study comparing the impact of similarity measures on cluster quality, because objectively evaluating cluster quality is difficult in itself. In practice, manually assigned category labels are usually used as baseline criteria for evaluating clusters. As a result, the clusters, which are generated in an unsupervised way, are compared to the pre-defined category structure, which is normally created by human experts. This kind of evaluation assumes that the objective of clustering is to replicate human thinking, so a clustering solution is good if the clusters are consistent with the manually created categories. However, in practice datasets often come without any manually created categories and this is the exact point where clustering can help. The rest of this section first describes the characteristics of the datasets, then explains the evaluation measures, and finally presents and analyzes the experiment results.

6.1 Dataset

This work experiments with two bench mark datasets "Reuters 21578 distribution 1.0" and Classic dataset collected from uci.kdd repositories. The Reuters-21578 collection is distributed in 22 files. Each of the first 21 files (reut2-000.sgm through reut2-020.sgm) contain 1000 documents, while the last (reut2-021.sgm) contains 578 documents. Documents were marked up with SGML tags, and a corresponding SGML DTD was produced, so that the boundaries of important sections of documents are unambiguous. Each REUTERS tag contains explicit specifications of the values of attributes such as TOPICS, LEWISSPLIT, CGISPLIT, OLDID, and NEWID. These attributes are meant to identify documents and groups of documents. Eg: <TOPICS>,</TOPICS>, <PLACES>, </PLACES>, <BODY>,</BODY>. Each will be delimited by the tags <D> and </D>. There are 5 categories Exchanges, Organizations, People, Places and Topics in the Reuters dataset and each category has again sub categories in total 672 sub categories. We have collected the TOPICS and PLACES category sets to form the dataset. The TOPICS category set contains 135 categories and PLACES category set contains 175 categories. From these documents we collect the valid text data of each category by extracting the text which is in between <BODY> ,</BODY> and placed in a text document and named it according to topic and place.

Classic dataset consists of four different collections CACM, CISI, CRAN and MED. We have considered 800 documents of the total 7095 documents.

In these datasets, some of the documents consists single word only, so it is meaningless to take such documents for document dataset. For eliminating these invalid documents we apply file reduction on each category, which returns the documents that supports mean length of each category. For file reduction we construct the Boolean matrices of all documents by category wise and calculate mean length of each category and removed the documents from the dataset which doesn't support mean length. By this we got valid documents. From these valid documents we have collected 800 documents of four categories each. From Reuters we have considered 200 documents of each category (ACQ, EARN of TOPICS

category and UK, USA, of PLACES category) totaling to 800 documents and from classic dataset 200 documents of each category again totaling to 800 documents.

6.2 Pre-Processing

Preprocessing consists of steps that take as input a plain text document and output a set of tokens (which can be single terms or n-grams) that are to be included in the vector model. In this work we performed Removal of stop word and stemming and built vector space model. Here we have pruned words that appear with very low frequency throughout the corpus with the assumption that these words, even if they had any discriminating power, would form too small clusters to be useful. Words which occur frequently are also removed.

6.3 Evaluation

For clustering, two measures of cluster "goodness" or quality are used. One type of measure allows us to compare different sets of clusters without reference to external knowledge and is called an internal quality measure. As mentioned in the previous section, we will use a measure of "overall similarity" based on the pair wise similarity of documents in a cluster. The other type of measures lets us evaluate how well the clustering is working by comparing the groups produced by the clustering techniques to known classes. This type of measure is called an external quality measure. One of the external quality measures is entropy, which provides a measure of "goodness" for un-nested clusters or for the clusters at one level of a Hierarchical clustering.

We use entropy as a measure of quality of the clusters (with the caveat that the best entropy is obtained when each cluster contains exactly one data point). Let CS be a clustering solution. For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the "probability" that a member of cluster j belongs to class i. Then using this class distribution, the entropy of each cluster j is calculated using the standard formula

$$E_j = -\sum_i p_{ij} \log(p_{ij})$$

where the sum is taken over all classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster:

$$E_{CS} = \sum_{j=1}^{m} \frac{n_j * E_j}{n}$$

where n_j is the size of cluster j, m is the number of clusters, and n is the total number of data points.

6.4 Results Analysis

In this work seed points are statically chosen, but efficiency can be improved if seeds selected are random or run the code more than once to check the efficiency. As shown in Tables 1a,1b and Tables 2a 2b, Euclidean distance performs worst while the performance of other measures is quite similar. From our results it is observed that Boolean representation with Pearson measure has non-zero clusters. Hence the overall entropy for Boolean representation table shows NAN values for other measures as some of the clusters are empty. On an average, the Jaccard and Pearson measures are slightly better in generating more coherent clusters, which means the clusters have lower entropy scores. Table 3a shows one partition as generated by the Boolean Pearson measure using Reuters dataset, and Table 3b shows one partition as generated by the TF-IDF Jaccard Coefficient measure using Classic dataset which has the lowest entropy value.

Table 1a: Entropy Results of Different Vector Space Representations Using Reuters dataset

	Cosine	Jaccard	Euclidean	Pearson
Boolean	NAN	NAN	NAN	0.33
Freq. Count	0.36	0.36	0.42	0.38
TF-IDF	0.36	0.38	0.42	0.37

Table 2b: Entropy Results of Different Vector Space Representations Using Classic dataset

	Cosine	Jaccard	Euclidean	Pearson
Boolean	NaN	NaN	NaN	0.06
Freq. Count	0.16	0.12	0.30	0.06
TF-IDF	0.06	0.07	0.30	0.06

Similar as above, the Euclidean distance is again proved to be an ineffective metric for modeling the similarity between documents. The Jaccard and Pearson's coefficient tend to outperform the cosine similarity.

Table 3a: TF-IDF Entropy Results using Reuters dataset

	Cosine	Jaccard	Euclidean	Pearson
Clusters[0]	0.41	0.16	0.38	0.41
Clusters[1]	0.33	0.38	0.44	0.33
Clusters[2]	0.26	0.40	0.40	0.28
Clusters[3]	0.31	0.16	0.42	0.30

Table 4b: TF-IDF Entropy Results using Classic dataset

	cosine	jaccard	euclidean	pearson
Clusters[0]	0.05	0.01	0.30	0.01
Clusters[1]	0.01	0.08	0.30	0.04
Clusters[2]	0.06	0.07	0.30	0.07
Clusters[3]	0.13	0.11	0.00	0.10

Table 5a: Clustering Results from Boolean Pearson Correlation Measure using Reuters dataset

	ACQ	EARN	UK	USA	Label
Cluster[0]	173	71	64	8	ACQ
Cluster[1]	18	12	107	57	UK
Cluster[2]	8	115	15	12	EARN
Cluster[3]	1	2	14	123	USA

Table 6b: Clustering Results from TFIDF Jaccard Measure using Classic dataset

	Cac	Cis	Cra	Med	Label
Cluster[0]	0	1	0	166	Med
Cluster[1]	8	5	199	30	CRA
Cluster[2]	3	166	0	4	CIS
Cluster[3]	189	28	1	0	CAC

We have used the clustering accuracy as a measure of a clustering result. Clustering accuracy r is defined as

$$r = \frac{\sum_{i=1}^{n} a_i}{n}$$

where a_i is the number of instances occurring in both cluster i and its corresponding class and n is the number of instances in the dataset. The clustering accuracy is more for TF-IDF representation with Pearsons and Jaccard coefficient measures. The classic dataset has shown above 94 percent accuracy.

7. CONCLUSIONS AND FUTURE WORK

In this study we found that all the measures have significant effect on Partitional clustering of text documents except for the Euclidean distance measurer. Pearson correlation coefficient is slightly better as the resulting clustering solutions are more balanced and is nearer to the manually created categories. The Jaccard and Pearson coefficient measures find more coherent clusters. Considering the type of cluster analysis involved in this study, we can see that there are three components that affect the final results—representation of the documents, distance or similarity measures considered, and the clustering algorithm itself. In our future work our intension is to apply semantics knowledge to the document representations to represent relationships between terms and study the effect of these similarity measures exhaustively.

REFERENCES

[1] C. J. Van Rijsbergen, (1989), Information Retrieval, Buttersworth, London, Second Edition.
[2] G. Kowalski, Information Retrieval Systems – Theory and Implementation, Kluwer Academic Publishers, 1997.
[3] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey, Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections, SIGIR '92, Pages 318 – 329, 1992.
[4] O. Zamir, O. Etzioni, O. Madani, R.M. Karp, Fast and Intuitive Clustering of Web Documents, KDD '97, Pages 287-290, 1997.
[5] D. Koller and M. Sahami, Hierarchically classifying documents using very few words, Proceedings of the 14th International Conference on Machine Learning (ML), pp. 170-178, 1997.
[6] G. Salton. Automatic Text Processing. Addison-Wesley, New York, 1989.
[7] M. Steinbach, G. Karypis, and V. Kumar. A Comparison of Document Clustering Techniques. In KDD Workshop on Text Mining, 2000.
[8] D. R. Cutting, J. O. Pedersen, D. R. Karger, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In Proceedings of the ACM SIGIR, 1992.
[9] B. Larsen and C. Aone. Fast and Effective Text Mining using Linear-time Document Clustering. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999.
[10] D. Arthur and S. Vassilvitskii. k-means++ the advantages of careful seeding. In Symposium on Discrete Algorithms, 2007.