Stack Overflow Data Mining

Sai Sathvick Chirakala
Computer Science
Texas Tech Unversity
sai-sathvick.chirakala@ttu.edu

Maitreyi Naik
Computer Science
Texas Tech Unversity
maitreyi.naik@ttu.edu

Nikhila Ratakonda
Computer Science
Texas Tech Unversity
nikhila.ratakonda@ttu.edu

***Abstract*—With the inception of World Wide Web, the amount of data present on the internet is tremendous. A large number of dataset repositories, catalogs and portals are emerging in the science and government realms. Once a large number of datasets are published on such data portals, the question arises how to retrieve datasets satisfying an information need. This makes the task of navigating through this enormous amount of data quite difficult for the user. In this work, we present a dataset which provides information on the questions that are posted in stack overflow, which is little bit noisy. Stackoverflow-data-idf.json file has 20000 questions posted and 19 columns. 19 columns include post title, body, tags, dates and other media.**

***Keywords*—IR, TF, IDF, TF-IDF**

The remainder of this paper is organized as follows: Introduction is explained in section 1; Procedures is in Section 2; Section 3 describes the results and discussions; section 4 gives the conclusions.

I. Introduction

Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers). It is the activity of obtaining information resources relevant to the information need from a collection of information system.

These days we frequently think first of web search, but there are many other cases:

• E-mail search
• Searching your laptop
• Corporate knowledge bases • Legal information retrieval

And the goal is to retrieve documents with information that is relevant to the user's information need and helps the user complete a task

II. Procedure and Related Work

A. *Review the Data Set Stage*

A Stack overflow dataset is taken, which is a little noisy. Stackoverflow-data-idf.json file has 20000 questions posted and 19 columns. The columns include post title, body, tags, dates and other media. After preprocessing the data, total terms left are 666637.

B. *Methodology*

First, we need to do the preprocessing of dataset, i.e. removing stop words and doing text normalization the following from the dataset. See the flow chart.

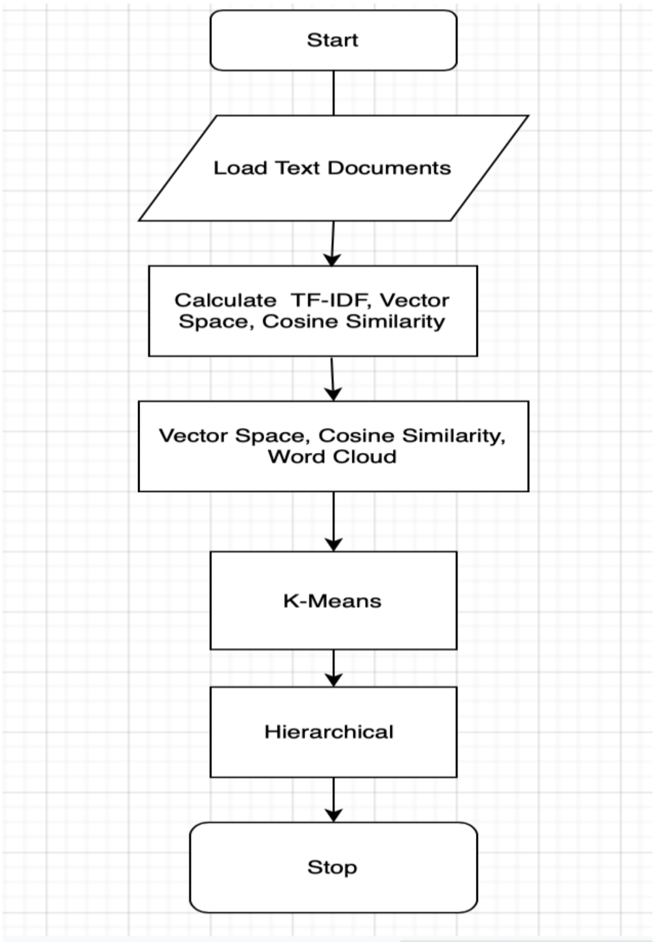

Start
Load Text Documents
Calculate TF-IDF, Vector Space, Cosine Similarity
Vector Space, Cosine Similarity, Word Cloud
K-Means
Hierarchical
Stop

Fig.1.Flow Chart

2.1. Pre-Processing of the Data

i. **Stop words:** These are basically a set of commonly used words in any language [1]
ii. **Stemming: It** is the process of reducing a word to its word **stem** that affixes to suffixes and prefixes or to the roots of words known as a lemma [2]
iii. **Lemmatization: It** usually refers to doing things properly with the use of a vocabulary and morphological analysis of words [3]
iv. **Tokenize:** It is the process of tokenizing or splitting a string, text into a list of tokens [4]

This method is called preprocessing.

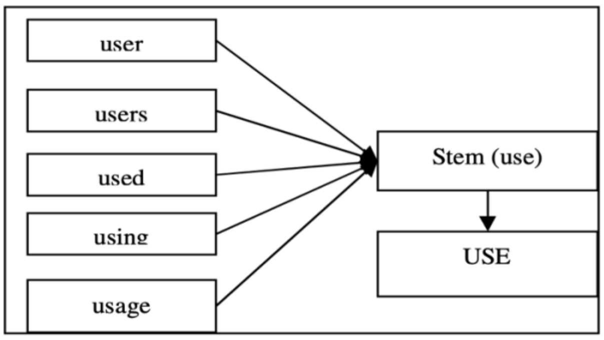


Fig.2. Stemming

2.2. TF-IDF

After the preprocessing we will define the query which is related to the document and calculate the term frequency, inverse document frequency.

i. **Term Frequency: TF** (**Term Frequency**) measures the **frequency** of a word in a document. TF = (Number of time the word occurs in the text) / (Total number of words in text). [5]

ii. **Inverse Document Frequency**: IDF = (Total number of documents / Number of documents with word t in it). [6]

Thus, the TF-IDF is the product of TF and IDF: [7]

TF-IDF = TF * IDF

2.3. Vector Space

A vector space model is an algebraic model, involving two steps, in first step we represent the text documents into vector of words and in second step we transform to numerical format so that we can apply any text mining techniques such as information retrieval, information extraction, information, filtering etc. [8]

2.4. Document similarity

Mathematically, closeness between two vectors is calculated by calculating the cosine angle between two vectors. In similar lines, we can calculate cosine angle between each document vector and the query vector to find its closeness. To find relevant document to the query term, we may calculate the similarity score between each document vector and the query term vector by applying cosine similarity. Finally, whichever documents having high similarity scores will be considered as relevant documents to the query term. [9]

When we plot the term document matrix, each document vector represents a point in the vector space. In the below example query, Document 1 and Document 2 represent 3 points in the vector space. We can now compare the query with each of the document by calculating the cosine angle between them. Fig.3.

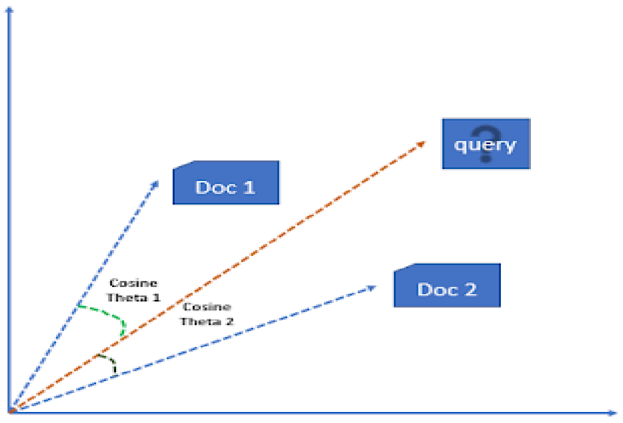


Fig. 3. cosine similarity

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \, \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|}$$

2.5. k-Means Clustering

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. It is popular for cluster analysis in data mining. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, Better Euclidean solutions can be found using k-medians and k-medoids.

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids.

It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

2.6. Hierarchical clustering

Clustering is basically a technique that groups similar data points such that the points in the same group are more similar to each other than the points in the other groups. The group of similar data points is called a **Cluster.**

Hierarchical clustering is one of the popular and easy to understand clustering technique. This clustering technique is divided into two types:

- Agglomerative
- Divisive

Agglomerative Hierarchical clustering Technique: In this technique, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.

The basic algorithm of Agglomerative is straight forward.

- Compute the proximity matrix
- Let each data point be a cluster
- Repeat: Merge the two closest clusters and update the proximity matrix
- Until only a single cluster remains

The Hierarchical clustering Technique can be visualized using a **Dendrogram.** A **Dendrogram** is a tree-like diagram that records the sequences of merges or splits.

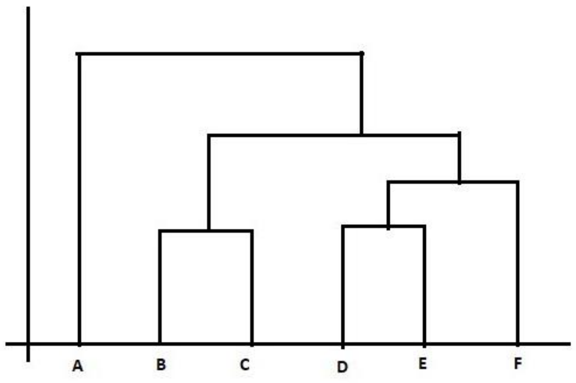


Fig.4. Dendrogram

Divisive Hierarchical clustering Technique: Divisive Hierarchical clustering is exactly the opposite of the **Agglomerative Hierarchical clustering.** In Divisive Hierarchical clustering, all the data points are considered as a single cluster and in each iteration, the data points are separated from the cluster which are not similar. Each data point which is separated is considered as an individual cluster. In the end, n clusters will be left.
Calculating the similarity between two clusters is important to merge or divide the clusters. There are certain approaches which are used to calculate the similarity between two clusters:

- Min
- Max
- Group average
- Distance between centroids

III. RESULTS

i. Tf-idf Score

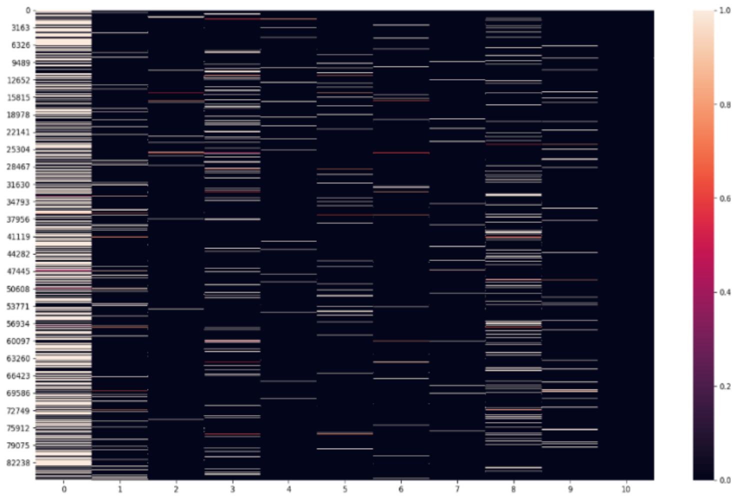


FIG.5. TF-IDF SCORE

ii. VECTOR SPACE HEAT MAP

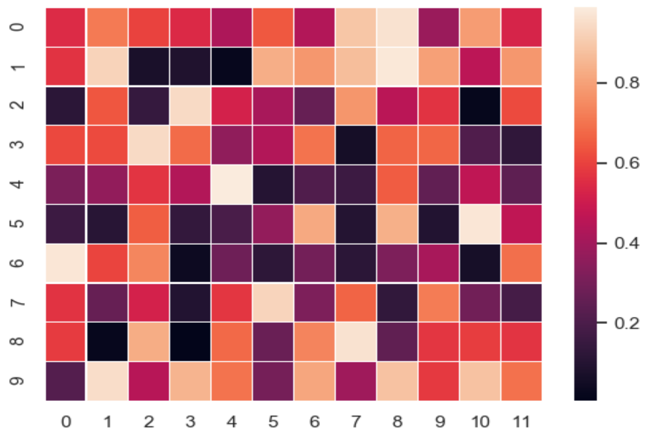


Fig. 6. Vector Space

iii. Document Similarity

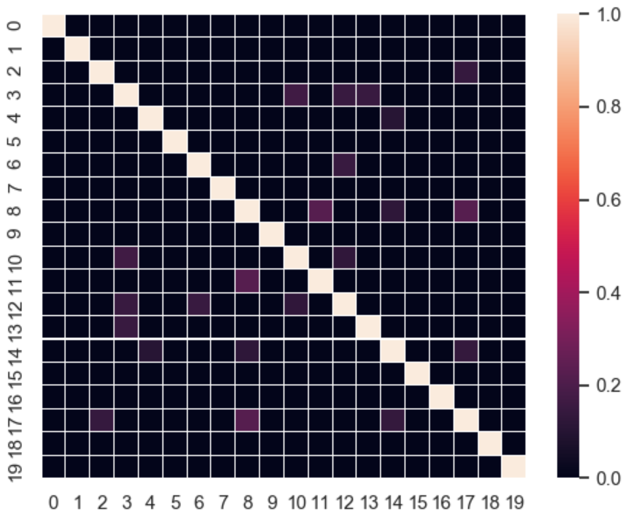
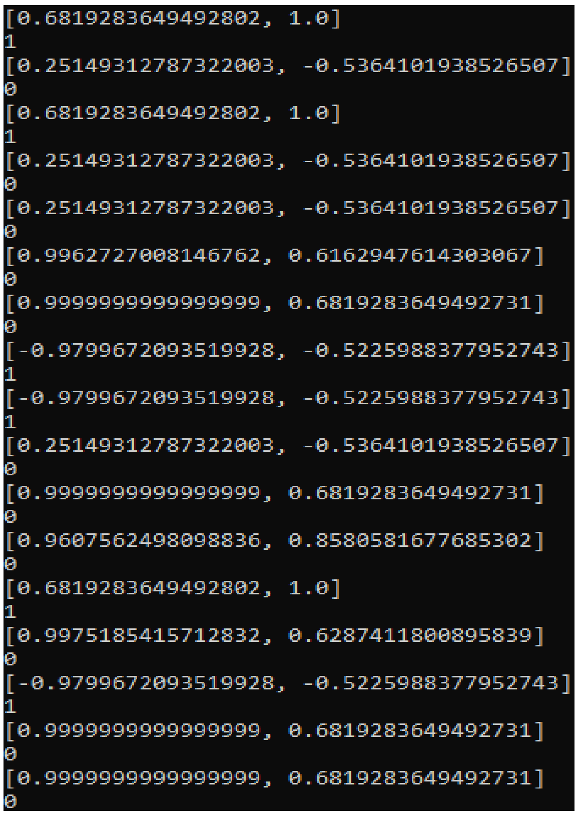


Fig. 7. Cosine Similarity

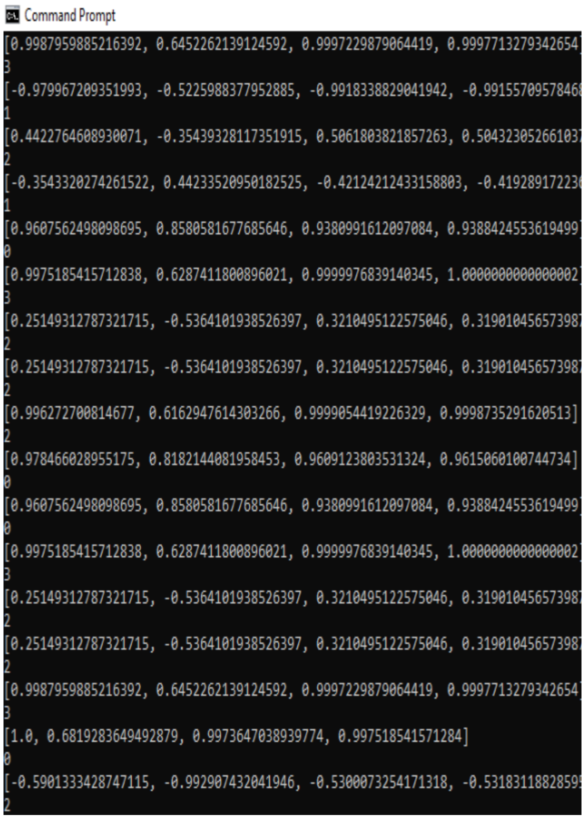
iv. K-means cluster for n=2



```
[0.6819283649492802, 1.0]
1
[0.25149312787322003, -0.5364101938526507]
0
[0.6819283649492802, 1.0]
1
[0.25149312787322003, -0.5364101938526507]
0
[0.25149312787322003, -0.5364101938526507]
0
[0.9962727008146762, 0.6162947614303067]
0
[0.9999999999999999, 0.6819283649492731]
0
[-0.9799672093519928, -0.5225988377952743]
1
[-0.9799672093519928, -0.5225988377952743]
1
[0.25149312787322003, -0.5364101938526507]
0
[0.9999999999999999, 0.6819283649492731]
0
[0.9607562498098836, 0.8580581677685302]
0
[0.6819283649492802, 1.0]
1
[0.9975185415712832, 0.6287411800895839]
0
[-0.9799672093519928, -0.5225988377952743]
1
[0.9999999999999999, 0.6819283649492731]
0
[0.9999999999999999, 0.6819283649492731]
0
```

Fig. 8. K-means cluster for n=2

v. K-means for n=4



```
Command Prompt
[0.9987959885216392, 0.6452262139124592, 0.9997229879064419, 0.9997713279342654]
3
[-0.979967209351993, -0.5225988377952885, -0.9918338829041942, -0.99155709578468
1
[0.4422764608930071, -0.35439328117351915, 0.5061803821857263, 0.504323052661037
2
[-0.3543320274261522, 0.44233520950182525, -0.42124212433158803, -0.41928917223
1
[0.9607562498098695, 0.8580581677685646, 0.9380991612097084, 0.9388424553619499]
0
[0.9975185415712838, 0.6287411800896021, 0.9999976839140345, 1.0000000000000002]
3
[0.25149312787321715, -0.5364101938526397, 0.3210495122575046, 0.319010456573987
2
[0.25149312787321715, -0.5364101938526397, 0.3210495122575046, 0.319010456573987
2
[0.996272700814677, 0.6162947614303266, 0.9999054419226329, 0.9998735291620513]
2
[0.978466028955175, 0.8182144081958453, 0.9609123803531324, 0.9615060100744734]
0
[0.9607562498098695, 0.8580581677685646, 0.9380991612097084, 0.9388424553619499]
0
[0.9975185415712838, 0.6287411800896021, 0.9999976839140345, 1.0000000000000002]
3
[0.25149312787321715, -0.5364101938526397, 0.3210495122575046, 0.319010456573987
2
[0.25149312787321715, -0.5364101938526397, 0.3210495122575046, 0.319010456573987
2
[0.9987959885216392, 0.6452262139124592, 0.9997229879064419, 0.9997713279342654]
3
[1.0, 0.6819283649492879, 0.9973647038939774, 0.997518541571284]
0
[-0.5901333428747115, -0.992907432041946, -0.5300073254171318, -0.53183118828595
2
```

Fig. 9. K-means cluster for n=4

vii. Hierarichal clustering

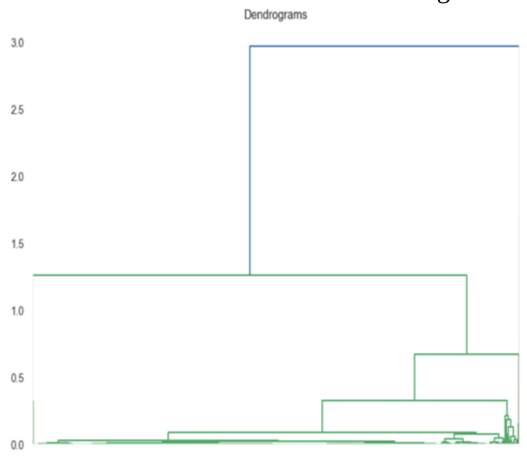


Fig. 10(a). Dendrogram

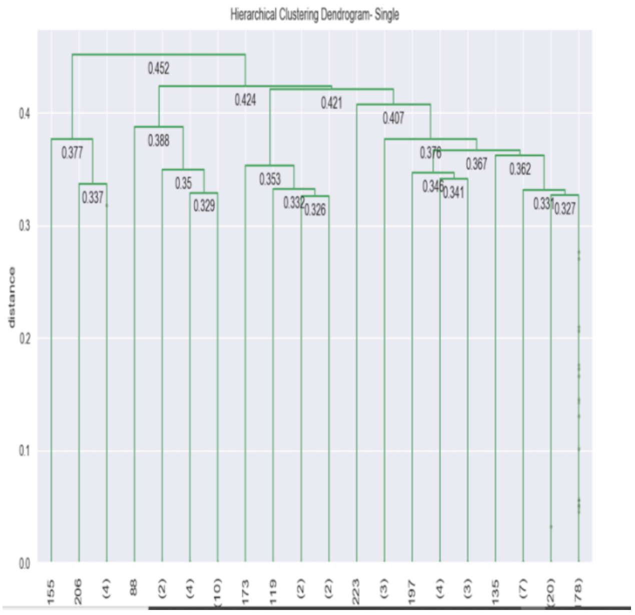


Fig. 10(b). Single

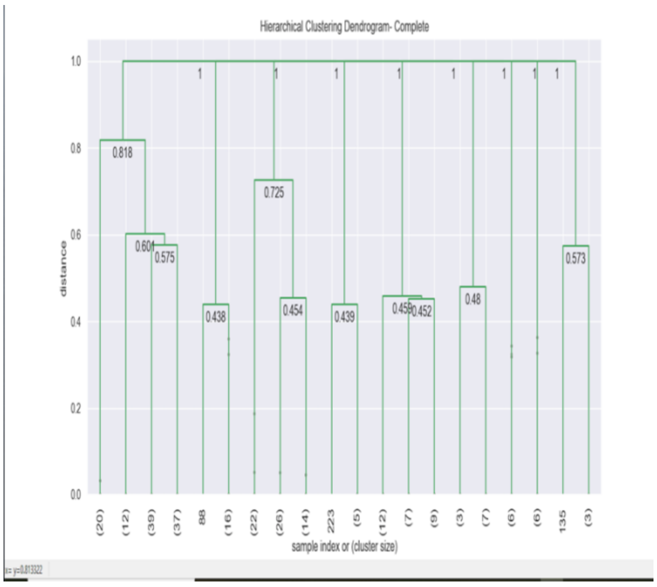


Fig. 10(c). Complete

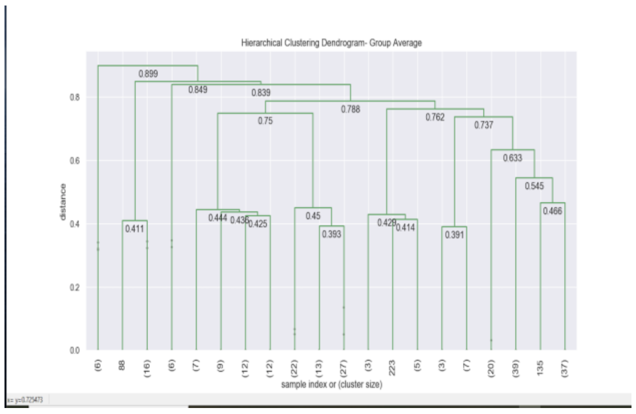


Fig. 10(d). Group Average

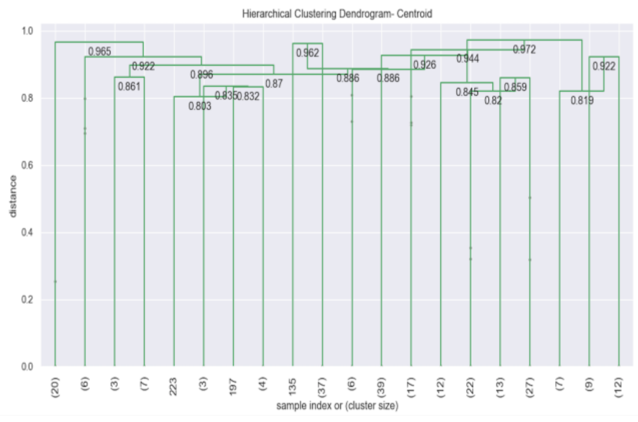


Fig. 10(e). Centroid

IV. CONCLUSION

We have seen that TF-IDF is an efficient and simple algorithm for matching words in a query to documents that are relevant to that query. From the data collected, we see that TF-IDF returns documents that are highly relevant to a particular query. If a user were to input a query for a particular topic, TF-IDF can find documents that contain relevant information on the query. Furthermore, encoding TF-IDF is straightforward, making it ideal for forming the basis for more complicated algorithms and query retrieval systems

Despite its strength, TF-IDF has its limitations. In terms of synonyms, notice that TF-IDF does not make the jump to the relationship between words. Since TF-IDF is merely a staple benchmark, numerous algorithms have surfaced that take the program to the next level.

Enhancing the already powerful TF-IDF algorithm and using K-means and hierarchical clustering would increase the success of query retrieval systems, which have quickly risen to become a key element of present global information exchange.

REFERENCES

[1] Wilbur, W. John, and Karl Sirotkin. "The automatic identification of stop words." *Journal of information science*18.1 (1992): 45-55.
[2] Lovins, Julie Beth. "Development of a stemming algorithm." *Mech. Translate. & Comp. Linguistics* 11.1-2 (1968): 22-31.
[3] Plisson, Joël, Nada Lavrac, and Dunja Mladenic. "A rule-based approach to word lemmatization. " *Proceedings of IS*. Vol. 3. 2004.
[4] Singh, Vikram, and Balwinder Saini. "An Effective tokenization algorithm for information retrieval systems." *Department of Computer Engineering, National Institute of Technology Kurukshetra, Haryana, India* (2014).
[5] Xia, Tian, and Yanmei Chai. "An improvement to TF-IDF: Term Distribution based Term Weight Algorithm." *JSW* 6.3 (2011): 413-420.
[6] Robertson S. Understanding inverse document frequency: on theoretical arguments for IDF. Journal of documentation. 2004 Oct 1.
[7] Ramos J. Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning 2003 Dec 3 (Vol. 242, pp. 133-142)
[8] Salton, Gerard, Anita Wong, and Chung-Shu Yang. "A vector space model for automatic indexing." *Communications of the ACM* 18.11 (1975): 613-620.
[9] Muflikhah, L., & Baharudin, B. (2009, November). Document clustering using concept space and cosine similarity measurement. In *2009 International Conference on Computer Technology and Development* (Vol. 1, pp. 58-62). IEEE.