**Online Medical Data Clustering for Search Result Diversification**

Avinesh Mohanaranjan: 216 325 854

Dawit Gulta: 219 251 156

Javier Guillen: 214 933 519

Richard Nguyen: 215 751 324

Vatsal Rajyaguru: 216 977 449

**York University**

**ITEC4305 M: Web Mining Assignment 2 Report**

**Dr. Jimmy Huang**

**March 14, 2023**

**Introduction**

The data set given is a ranked passage retrieval result file “output-format-york07ga1.txt” formatted as follows: topic number, document ID, rank of the passage, system assigned score, byte offset, length of the passage in bytes, tag ID, and passages. The file “top-passages-york07gal.txt” contains up to 1,000 passages per topic that correspond to the "output-format-york07ga1.txt" passage retrieval file. Each record in the file “top-passages-york07gal.txt” is formatted as follows: topic ID, passage ID (concatenation of data from document ID to tag ID separated by underscores) and its passage content. We will first preprocess the data, implement a clustering program to group retrieved passages for each topic and re-rank the original retrieved passages to improve aspect search performance. The re-ranked results will need to produce an output containing the same format as “output-format-york07ga1.txt.”

**Description of Online Medical Data-Clustering Program**

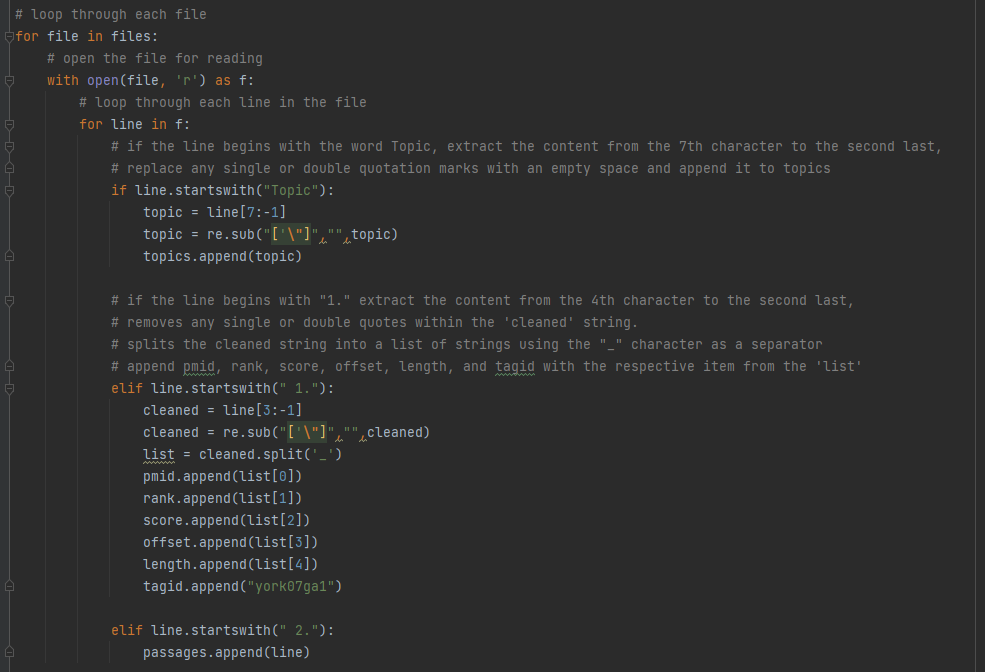
Our python code has two files, Preprocessing.py to preprocess the data and Clustering.py to group retrieved passages.

The preprocessing python file defines functions to preprocess the text data and extracts the topic, passage ID, rank, score, offset, length, tag ID and passages. Stop words and improper punctuation are removed from the extracted passages. The cleaned passages are then converted into vectors using TF-IDF vectorization, and the top keywords for each passage are extracted. The extracted data and keywords are written to a CSV.

The clustering python file uses the KMeans algorithm to perform clustering on the set of passages extracted from the csv file. This algorithm was chosen because of its flexibility with large datasets and ability to handle high dimensional data such as text data. Healthcare is an example of high dimensional data where the number of features for a sample can be larger than the number of data points. The text data can be clustered based on the similarity of their TF-IDF features (Term Frequency-Inverse Document Frequency). TF-IDF is a statistical measure used to assess the significance of a word in a collection of documents. Based on the similarity of its features, Kmeans algorithm assigns each document to one of the K clusters. The resulting clusters can be used for further analysis or to identify patterns and trends in the text data. The text data extracted from results.csv is initially processed with a TF-IDF Vectorizer which fits and converts the text data into a dense matrix. The KMeans algorithm is then implemented on the TF-IDF matrix, and the resulting cluster labels are appended to the original dataframe as a new column. The clustered data is saved to a new CSV file named ‘cluster\_file.csv.’ The PCA algorithm is used to reduce the TF-IDF matrix's dimensionality to 2 components, which are shown as a scatter plot. By reducing the number of dimensions from the original high-dimensional space to a lower-dimensional space, this can help with visualization and further analysis. Each point is coloured based on its cluster label and a legend is added to the plot. The scatter plot produced represents the two-dimensional distribution of the text data as well as the clustering pattern.

**Description of Implementation**

When preprocessing libraries like csv, re, stopwords from nltk.corpus, string, and TfidfVectorizer from sklearn.feature\_extraction.text were imported. Empty lists for topics, pmid, rank, score, offset, length, tagid, and passages are created. The open() function is used to open the input file "top-passages-york07ga1.txt" and then use the file to loop three times. If the line begins with the word "Topic", it extracts the content after the 7th character until the second last character, removes any single or double quotation marks within the content, and appends it to the topics list. If the line starts with "1." it extracts the content after the third character until the second last character, removes any single or double quotation marks within the content, splits the cleaned string into a list of strings using the "\_" character as a separator, and appends pmid, rank, score, offset, length, and tagid to the respective item from the list. If the line starts with "2.", it appends the line to the passages list.

Stop words and punctuation are removed from the text using the stop\_removal() function. The preprocess() function applies stop\_removal() to each passage in the passages list and stores the result in cleaned\_passages. A TfidfVectorizer object is defined containing the following parameters:

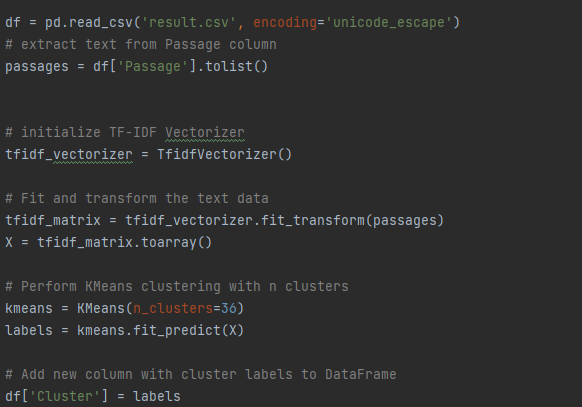
* lowercase=True: convert all text to lowercase.
* max features=100: set the maximum number of features to 100.
* max\_df=0.8: filter out words that appear in more than 80% of the documents.
* min df=5: exclude words found in less than 5 documents.
* ngram range=(1,3): separate phrases from a sentence using unigrams, bigrams, and trigrams
* exclude english: exclude English stop words



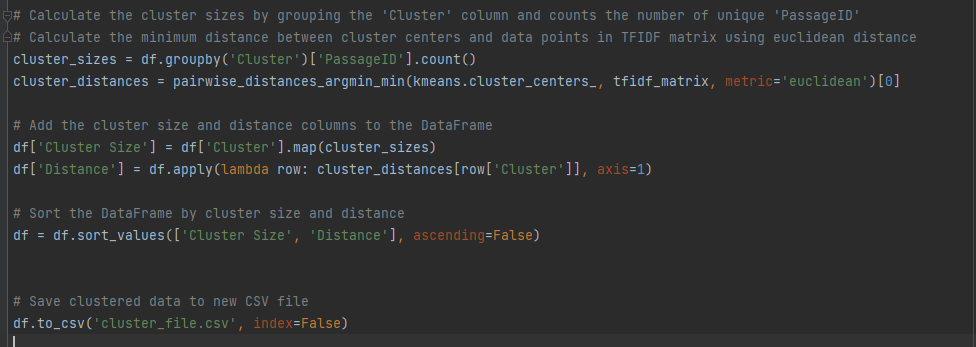
Using these parameters, the TfidfVectorizer object fits the cleaned\_passages according to the parameters and produces feature names and dense vectors. The dense vectors are converted to a denselist and keywords are extracted from that list that have non-zero values. The keywords are stored in all\_keywords. The results are written to a CSV file named "result.csv.” The csv.writer function is initialized to write a new row to the CSV file. A header row is created containing Subject, PassageID, Rank, Score, Offset, length, tagid, and Passage. A for loop is used to iterate across the range of values from 0 to the length of the topics list and values which are separated by a comma are written to the CSV file as a single row.



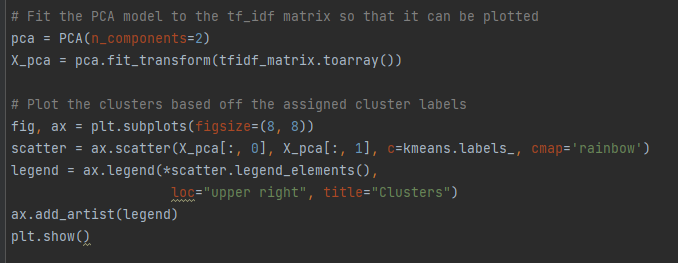
Clustering.py imports libraries numpy, pandas, scipy.sparse, matplotlib.pyplot, Kmeans and PCA from scikit-learn package, and TfidfVectorizer from sklearn.feature\_extraction.text. Dataframe df is declared to load the CSV file results.csv. ‘Passages’ extracts text from the passage column and converts it to a list to be used for feature extraction. TfidfVectorizer object is initialized to convert the text data into a feature matrix of TF-DIF scores. tfidf\_matrix fits the TfidfVectorizer to the ‘passages’ list creating a feature matrix of TF-DIF scores. Feature matrix is then converted to a dense matrix using the toarray() function. Kmeans object is initialized on the TF-IDF matrix (X) to perform clustering with 36 clusters. fit\_predict is executed to retrieve the cluster labels for each dataset. The cluster labels are added to a new column to the dataframe.



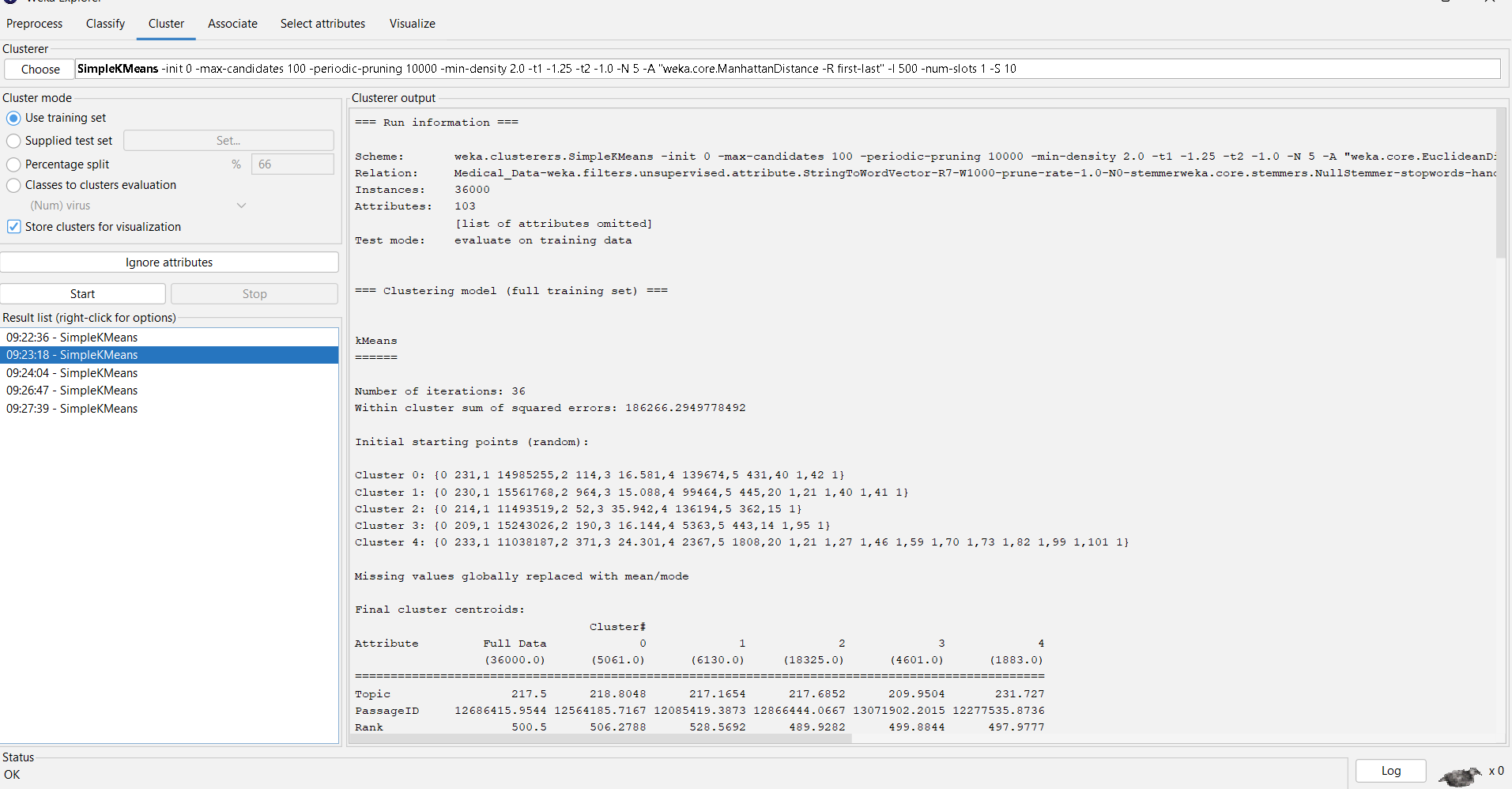
To implement reranking, Cluster sizes and distance are calculated based on the number of data points in each cluster and minimum distance between data points and cluster centers using euclidean distance. The dataframe is sorted in descending order based on cluster size. Then, the clustered data is stored to a new CSV file named ‘cluster\_file.csv’, without the index column.



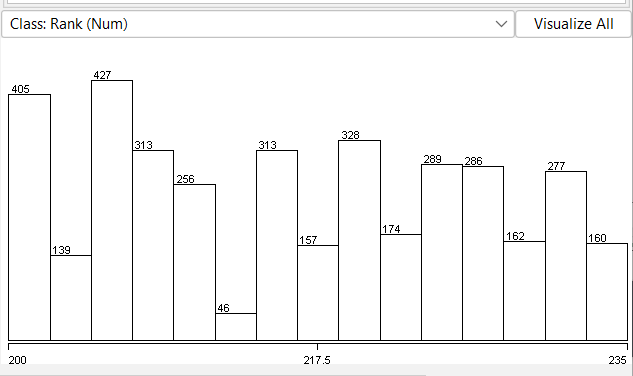
A PCA object with two principal components is initialized to fit the TF-DIF matrix. The fit\_transform method is executed to retrieve the PCA transformed feature matrix. A scatter plot is produced of the clustered data with the x and y coordinates defined by the PCA transformed feature matrix.



As an alternative, Weka is used to upload "result.csv” generated from our preprocessing code in python and group the retrieved passages for re-ranking purposes. We preprocessed the dataset before performing clustering. This involved removing stopwords, stemming and tokenizing the text. To remove stopwords, the text file was filtered by ‘StringToWordVector” applied remove stopwords. K-means and hierarchical clustering are used to perform clustering on the preprocessed dataset. We were able to re-rank the results after clustering based on the similarity between the documents.



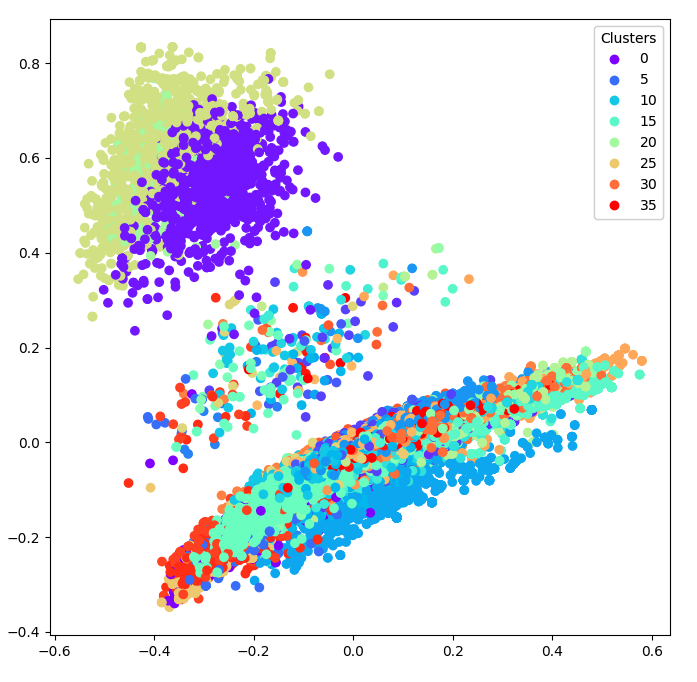
Weka uses Ranks attributes by their individual evaluations. Use in conjunction with attribute evaluators (ReliefF, GainRatio, Entropy etc).





**Analysis**

By plotting the clusters onto a graph, it helps give us a visualization of the various clusters. This would end up helping us with reranking as we can validate whether the graph would correlate to the reranking based on the cluster size. Issues we faced during the assignment were understanding how to re-rank the data with python. As an alternative, we used the "result.csv” generated from our preprocessing code in python and uploaded it to Weka.

****

**Conclusion**

The focus of this project was to obtain experience in preprocessing, grouping online medical data and re-ranking the output retrieval results. Preprocessing is mandatory for converting the raw data into a format that can be analyzed by machine learning algorithms and data visualization. Kmeans algorithm was chosen because of its flexibility with large datasets and ability to handle high dimensional data such as medical data. PCA is used to reduce the dimensions of the TF-IDF matrix to two dimensions for plotting the results of the clustering analysis and help us visualize the data. Weka is used to upload "result.csv” generated from our preprocessing code in python and use K-means and hierarchical clustering on the preprocessed dataset. We were able to re-rank the results after clustering based on the similarity between the documents.