FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own work, of work of another person, whether or not such work has been published" (Regulations Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta).

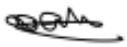
~~I~~/ We*, the undersigned, declare that the [assignment / ~~Assigned Practical Task report / Final Year Project report~~] submitted is ~~my~~/ our* work, except where acknowledged and referenced.

~~I~~/ We* understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

* Delete as appropriate.

(N.B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Owen Agius	
Student Name	Signature
Paolo Bezzina	
Student Name	Signature
Student Name	Signature
Student Name	Signature

UBSCITHARIFT	Web Intelligence Group Assignment
Course Code	Title of work submitted

14/02/2021

Date

Text Analysis on the Enron Dataset

Rather than working with the whole dataset, we decided it would make more sense to work on a fraction of the dataset. This is because it would take a large amount of time to process the whole dataset and so by using just a small part of it, you would still get results, but without wasting time waiting for the set to compile. However, if the whole dataset is used, the program would still function normally. This smaller dataset was called “maildirtest”

The creation of the JSON file essential for the D3 Visualisations is created by the functions in the ‘parsingEnron’ python executable. The class starts off by creating an object of type ‘Email’ in order to store the parsed data from the given dataset. The object accepts the email sender, recipients, and body as parameters.

Parsing

In order to access the emails themselves, the os library was used. The root directory “maildirtest”, which holds the dataset provided is access using the os.walk function which gives us three variables, ‘directory’, ‘subdirectory’ and ‘filenames’. The ‘filenames’, in a particular subdirectory are iterated over with iterator ‘filename’ in a nested for loop. The file is opened by indicating the current directory and the particular file name, and the parsed data is stored in a variable ‘data’.

The sender of the emails is found with the use of the *email.parser* python library. An instance of the library was created where, in this case, ‘from’ was passed as an argument to return the ‘from’ section of the email. A similar operation is carried out when reading the ‘to’ section of the email where the result of the *emailParser* instance is saved into the recipients’ variable. The same operation is carried out for the ‘bcc’ and the ‘cc’ fields. Since all the ‘to’, ‘bcc’ and ‘cc’ fields indicate a recipient, these are all added to the same list of recipients. Similarly, the body of the email is parsed using the emailParsers’ ‘get_payload(), which returns the message of the email.

An object of type Email is instantiated and appended to the list of all emails in the dataset. Similarly, a list of all the email addresses is created, which includes all the email addresses which sent or received emails. Afterwards, a dictionary is created of all the combined documents passed between two distinct people by iterating through the emails and recipients lists.

Pre-processing

Pre-processing entails the carrying out of arranging the data to be valid for TF-IDF weighting and Cosine Similarity. Pre-processing consists of tokenisation, case folding, stop word removal, stemming and symbol removal. Tokenisation is the process of breaking down a sentence or paragraph into a list of words. Case folding converts all the data passed into lowercased symbols. A pre set value of stop words, such as “and”, “then” and “is”, are removed from the dataset through the execution of stop word removal, this selection removes the data which is not more important or relevant to the document. Stemming is the process of setting a particular word to its most basic form, for example: “Stemming” or “Stemmed” are outputted to be “Stem”. Symbols usually carry negligible importance in a document, therefore a variable symbol is iterated over and checks whether a term in data contains a symbol, if a symbol is found, the data without the symbol is returned and updated in the datset.

Calculating TF-IDF

Term Frequency – Inverse Document Frequency is a technique to quantify a word in documents. A weight is computed for each word which signifies the importance of the word in the document and ergo the dataset. This weight is then stored in a term by document matrix.

Firstly, the Document Frequency is calculated. The DF calculation function returns a dictionary of the number documents in which each word can be found. The calculation happens by iterating through the length of the dataset and the tokens of the dataset. The tokens are added to a temporary dictionary set in the document frequency function. Afterwards, the dictionary is iterated over, where each value in the dictionary is set to be the length of the same element.

The document values, keys and unique words are now set to be stored in a list, and a DF dictionary is calculated on the list of values. The dictionary *tfidf* is filled up by using an iteration

over the length of the dataset and the number of unique tokens as a nested for loop. The TF is calculated by dividing the counter of the current token by the length of the tokens. The DF is calculated by value to be the element in the dictionary of the current token. Meanwhile, the IDF can be described as the log of the length of the dataset divided by the DF. Therefore, the TFIDF is the TF multiplied by the IDF.

Finally, the algorithm gets all the documents in which a user X participates in, participation can be either sending or receiving. All the TF-IDF values of the documents have their averages stored and the average value is linked with user X. This process is repeated for all the users. The last command executed is that the dictionary is converted to a vector.

Calculating K- Means Clustering Coefficient

The clustering algorithm was tackled by firstly converting the dataset into a *numpy* array. Three points are initialized randomly from the dataset. These three numbers were used as indices and the data points of the indices were returned. The three numbers are our initial centroids, which are then converted to a two-dimensional array. Iterating through all the data points in the dataset, we calculate the distance from the current point to all of the three centroids. And then the point is assigned to the centroid with the smallest distance to it. Finally, the centroids are moved based on the mean of the data points.

Cosine Similarity

Instead of building our own cosine similarity functions, we opted to use the predefined cosine function in the *numpy* library. This is because, upon testing the dot product and normal multiplication were outputting the same values, which is clearly a runtime error, hence resulting in all of the cosine similarity values to be 1. So instead of settling and working with incorrect values, we used the predefined function to continue with the tackling of the task at hand.

Visualisations

As a way to show visualisations we have created a dashboard using html, css as well as javascript. In this dashboard a number of visualisations were chosen to be represented. A force-directed graph would show the network of all connected users. This means if 2 users corresponded, there would be an edge that connects both of them. A Bubble chart was chosen, in order to show which users communicated most, regardless of with whom the communication was carried out. Finally, each user would also have a keyword cloud, which would show the top 20 words a user uses in all their emails. The more the user uses a word, the larger it would appear in the cloud. This would be accessed by clicking on the user from the bubble chart. These three visualisations together would visualise the dataset enough to be understood just by looking at the charts rather than studying the numbers.

Division of Work

Paolo tackled most of Task 1 and Owen tackled most of Task 2 but when either met any problems, we consulted each other, hence participating in both tasks equally. It is also important to note that we checked in frequently on each other to ensure that both students are working and are grasping knowledge of the task at hand.