The Report of Understanding the Data of Antiquity

Jing Meng
29299675
jm4y17@soton.ac.uk

1 INTRODUCTION

The aim of the project is to explore the data sets of 24 articles about Antiquity. Each article consists of loads of HTML files which are scanned by OCR. Hence, the first step is to extract the content from these HTML files. The subsequent steps are clean, feature extraction, feature engineering to explore the relations among documents based on features and finally clustering.

2 PRE-PROCESSING

The first step is to extract the words from the HTML file using *BeautifulSoup* library in python. There is a problem during the parsing process, which is a two words are split by lines . The built-in function *get_text* in *BeautifulSoup* merges the two words split by two adjacent rows with no delimiters. The new parsing method is to check whether the letters at the last of a line and the letters at the beginning of the next line combines an entire word by the built-in corpus *wordnet* in *nltk* package, if so, combine to a word, if not, split the two words by a space. The pre-processing can be devided as follows: (1). Remove the punctuation and all numbers, change all characters to low cases (2). Tokenize (3). Lemmatize (4). Remove all stopwords (5). Remove the words whose letters are less than 2. The point needs to point out is to use the lemmatizaton instead of stemming, because the stemming can not restore some words correctly, like plurals, e.g. cities → citi, leaves → leave, and preterite, e.g. said → said, remove the final letter 'e' of some words, e.g. people → peopl. The lemmatization function needs to indicate the part of speech of the word, thus, firstly, use the *pos_tag* function to gain the part of speech of the word, then as a parameter call the lemmatization function. The difference between stemming and lemmatization is lemmatization usually means to remove inflectional endings only and to return the base or dictionary form of a word, stemming refers to remove the ends of words including the removal of derivational affixes [5].

3 FEATURE EXTRACTION

3.1 Term Frequency.Inverse Document Frequency(TF.IDF)

The parameters given in the function *TfidfVectorizer* is $min\_df = 0.01, max\_df = 0.9, max\_features = 15000, analyzer =' word'$, $ngram\_range = (1, 1), use\_idf = 1, smooth\_idf = 1, sublinear\_tf = 1, stop\_words =' english'$. We can get the TF.IDF scores matrix of 24 documents with 15000 features. The 2D regular raster of the tf.idf score for the first 30 features is as follows. After the TF.IDF process, we can calculate the consine similarity based on the result of TF.IDF.

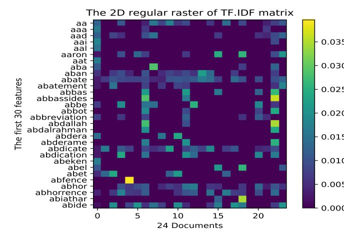
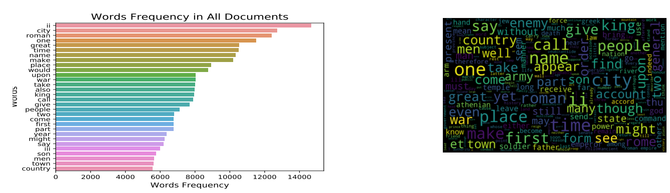


Figure 1: The 2D regular raster of TF.IDF matrix

4 FEATURE ENGINEERING

4.1 Word Frequency and Word Cloud

The occurrence of top 30 most frequently used words in all documents and word cloud figure are as follows. The statistics of the word frequency is just based on the term frequency not considering the inverse document frequency. Hence, the result is different from TF.IDF and it is the original and roughest way to explore the features of the data. From the figure, we can find the word with highest frequency is *ii* which is the roman number for chapter count. Other words like *roman*, *war* have the reflection of the content.



(a) The Top 30 Most Frequently Used Words (b) The Word Cloud

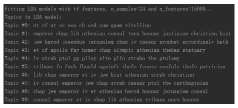
Figure 2: The Word Occurrence

4.2 Topic Modelling

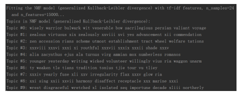
4.2.1 Non-Negative Matrix Factorisation(NMF). There are several types loss functions, e.g. frobenius, kullback-leibler, itakura-saito. We use the kullback-leibler loss function, the parameter of the function NMF is $n\_components = 10, random\_state = 1, beta\_loss =' kullback - leibler', solver =' mu', max\_iter = 1000, alpha = .1, l1\_ratio = .5$

4.2.2 Latent Dirichlet Allocation(LDA). The parameters for $LatentDirichletAllocation$ function is $n\_components = 10$, $learning\_method =' online', learning\_offset = 50., random\_state = 0$

The following lists the 10 topics and 10 words associated with each topic by NMF and LDA receptively:



(a) The NMF Topic Model



(b) The LDA Topic Model

Figure 3: The Topic Modelling

The figure shows the topics extracted by the two methods are different. The LDA more find nouns, the most frequently words are *emperor*, *jew*, *consul* and some people's names. Whereas, the NMF more discover adjectives, adverbs and roman numbers.

5 CLUSTERING

5.1 Hierarchical Clustering

We use $linkage\_matrix = ward(tf\_dif\_matrix)$, define the $linkage\_matrix$ using ward clustering pre-computed distances.

5.2 K-Means Clustering

The K-Means Clustering is to group data sets recursively by assigning to nearest centoid [3] [1]. Because K-Means algorithm needs to provide the number of cluster, there is a problem what is the optimal value for clusters. We use silhouette analysis [2] to gain the solution. The range of Silhouette coefficients is [-1,1]. The coefficients near +1 mean that the sample is far away from the neighboring clusters [2]. The 0 value indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values show that those samples could have been grouped to the wrong cluster [2]. The coefficients for different clusters are as follows. Thus, we choose 5 as the number of groups for clustering.

	2	3	4	5	6
Silhouette Score	0.0854	0.1139	0.1278	0.1477	0.1446

Table 1: The Silhouette Coefficients for Different Clusters

5.3 Mean Shift Clustering

The mean shift clustering is to cluster data samples recursively according to find the points in feature space with the highest density [3] [1]. The key parameter is kernel function and bandwidth. Different bandwidth impacting accuracy of probability density estimation [4]. We use $estimate\_bandwidth$ function to estimate the bandwidth, the parameters are $quantile = 0.2, n\_samples = 500$, for $MeanShift$ function, we set $bin\_seeding = True$, let the function $clustering.get\_bin\_seeds$ to get the centroid automatically.

The above clustering approaches, except K-Means need to provide the number of cluster, other twos determines the cluster number automatically.

6 VISUALISATION

Visualisation is for visulising the cluster of data sets from a high dimensional space to a lower dimensional space by some methods like Multidimensional Scaling.

6.1 Multidimensional Scaling(MDS)

MDS has two main types: Metric MDS matching distances and non-metric MDS matching rankings. There is no explicit mapping between them [3] [1]. The purpose of MDS is to minimise a stress function. We reduce the dimension to 2 for visualisation. The parameter of function MDS is $n\_components = 2, dissimilarity = "precomputed", random\_state = 1$. Here, we use $dist\_matrix = 1 - cosine\_similarity(tf\_idf\_matrix)$ for fitting.

The following figures show the visualisations in 2 dimensional spaces for Hierarchical, Mean Shift, and K-Means clustering.

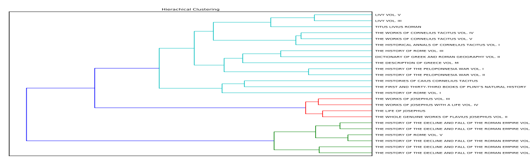
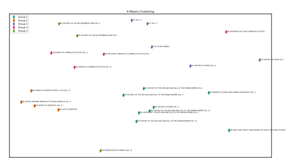
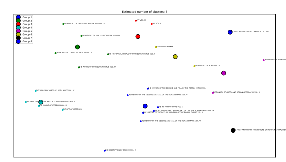


Figure 4: The MDS Reduction for Hierarchical Clustering



(a) The MDS Reduction for K-Means



(b) The MDS Reduction for Mean Shift

Figure 5: The MDS Reduction

In sum, the titles of these documents are stated above, the results show that hierarchical clustering groups the 24 documents to final 3 classes in generally. From the titles, we can get the clustering is correct basically. The content of the top(light green) group is basically about Greece and Tacitus, the middle group is about Josephus, the last one is almost about the history of Roman. The clustering for K-Means and Mean Shift is slightly different from Hierarchical. In general, the 3 groups whose content about Josephus, Tacitus, and The history of the Roman are the main groups, there are just more clusters for K-Means and Mean Shift. For the relatively far documents "THE DESCRIPTION OF GREECE", "THE FIRST AND THIRTY-THIRD BOOKS OF PLINYŚ NATURAL HISTORY", and "THE HISTORIES OF CAIUS CORNELIUS TACITUS", the three clustering algorithms have their own regulation. Obviously, it is also related the selection of relevant key parameters respectively as mentioned above.

7 CONCLUSIONS

In the project, we use some descriptive data mining techniques to explore the features and similarity among the data sets. From the analysis process, it can be concluded that the pre-processing decides the performance of the result, the parameters provided to each built-in function are the key factors of the final result.

REFERENCES

[1] A. Géron. 2017. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media. https://books.google.co.uk/books?id=bRpYDgAAQBAJ

[2] scikit-learn developers. 2018. Selecting the number of clusters with silhouette analysis on KMeans clustering. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py

[3] T. Segaran. 2007. *Programming Collective Intelligence: Building Smart Web 2.0 Applications.* O'Reilly Media. https://books.google.co.uk/books?id=fEsZ3Ey-Hq4C

[4] Burr Thomas. 2008. Pattern Recognition and Machine Learning Christopher M. Bishop. *J. Amer. Statist. Assoc.* 103, 482 (2008), 886. http://search.ebscohost.com/login.aspx?direct=true&db=edsjsr&AN=edsjsr.27640118&site=eds-live

[5] The Stanford University. 2009. Stemming and lemmatization. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html