RESEARCH SKILLS:
DATA PROCESSING ADVANCED
Group Assignment

Professor:	G.A. Chrupala
Date:	7th December 2018
Group number:	05
Günsu Şahin	[u557036]
Sema Karan	[u924823]
Maria Eira	[u900574]

CodaLab Account name: skaran (group05)

Experimental setup

The aim of this work was to develop an algorithm to recognize the similarities between different texts. Given a specific data set that contains 10000 questions, a model has been built to match the questions that contain the same information. Final error rate on the test data was …..? The process of achieving this score is elaborated below.

Preprocessing

For the implementation of this setup, several libraries have been used; namely “Pandas” for reading the data, “Scipy” to represent the data as sparse matrix, “SkLearn” for similarity and distance metrics and “NLTK” for text processing. The text processing was performed by the function “clean_text”. This function was developed to remove all the punctuation, remove stop words (stop words are the words that occur frequently hence not beneficial in terms of differentiating between texts), implementing the stemming and lemmatization techniques. The function can apply these methods individually or combined.

Data Representation

Words in the questions have been converted to a matrix of token counts with “CountVectorizer” function in the SkLearn library which uses Scipy to return a sparse matrix of the counts. Matrix of token counts were used to calculate term frequency- inverse document frequency (Tf-Idf), which shows the importance of a word for a given context (in this case the questions). Tf-Idf measures the significance of a word based on its occurrence in the text, multiplied by its inverse document frequency which looks at the occurrence of that particular word in the whole documents. Therefore, it creates a punishment mechanism for frequently occurring words in different documents which have no significant differentiating power. After the calculation of the Tf-Idfs, different distance metrics have been tried to calculate the distances between questions.

Distance/Similarity Metrics & Tuning

Cosine Similarity metric has been applied using SkLearn’s pairwise cosine similarity function. After measuring the distances and converting it in a sparse matrix, the matrix was converted to data frame to filter out the diagonal values (same question overlaps) and find the maximum cosine distance value. Finally, the question indexes from the original data has been added to the data frame to match the PARA_IDs with predicted (mapped) ids to calculate the error rate on training data.

Results & Discussion

Jaccard, Euclidean and Cosine Similarity metrics have been tested on the development data. The codes written to calculate Jaccard and Euclidean had a higher error rate than the Cosine Similarity. Additionally, literary reviews on the subject of text similarity have pointed out the Cosine Similarity metric as one of the best measures to use, therefore, we chose the last one.
Tuning has been made for the preprocessing phase, after observing that some of the punctuations was important for the meaning of the words, for example the signals “+” and “#” were essential to

distinguish the software language words “C++” from “C#” that are in some questions. Therefore, from the punctuation included in the “String” library, only following signs were removed: “!"$%&\'()*,-./:;<=>?@[\\]^_`{|}~”. However, punctuations were not further processed to avoid the potential risk of overfitting.
During the analysis of the stop words included in the NLTK corpus, some meaningful words like verbs (“be”, “have”, “do” etc.) have been found to be removed. From the results with and without stop words, it has been confirmed that the presence of these words was necessary to grasp the similarities between questions. Additionally, stemming alone proved to give a better accuracy than lemmatization and the combination of both techniques.
The following table shows a summary of the distance metrics tried and tuning methods applied:

Distance / Similarity Metrics	**Tunings**	**Error Rate**
Jaccard		-
Eucledian		0.4
Cosine Similarity	remove stopwords + tf-idf	0.150
Cosine Similarity	keep stopwords + tf-idf	0.130
Cosine Similarity	keep stopwords +remove punctuation+ stemming + tf-idf	0.049
Cosine Similarity	keep stopwords + lemmatization _wordnet_POS + tf-idf	0.100
Cosine Similarity	keep stopwords +stemming + lemmatization_wordnet_POS + tf-idf	0.050
Cosine Similarity	keep stopwords + lemmatization _TextBlob + tf-idf	0.110
Cosine Similarity	keep stopwords + stemming + tf-idf + remove !"#$%&\'()*,-./:;<=>?@[\\]^_`{|}~	0.0497
Cosine Similarity	keep stopwords + stemming + tf-idf + remove !"\'&(),-./:;<=>?[\\]^_`{|}~	0.05

As it’s emphasized in yellow, the lowest error rate on training data is gained by removing punctuations with stemming and transforming to tf-idf, so we used with this approach to apply test data and test error rate was 0.0599 on Codelab.

Group tasks

The assignment introduction was carried on by all the members of the group. The subsequent work was divided as the follows:

Group member	***Tasks***
Günsu Şahin	Vectorization, Tf-idf, Jaccard coefficient, report (until results & discussion)
Sema Karan	Cosine similarity, transforming output,error rate, preparing the final script
Maria Eira	Preprocessing, Euclidean distance, report (results & discussions)