Group 21 IPR Project 2 Report

Daniel Correia, Gonçalo Ramos, João Machado

Instituto Superior Técnico

1 Exercise 1

Regarding exercise 1, through a .txt serving as an input document, the keyphrases were extracted by the TfidfVectorizer with ngrams between 1 and 3 and without stopwords. We then develop a structure composed by the terms in each phrase. As our graph, for every co-occurrence between terms in the same sentence, we make the association between terms.To obtain the page-rank, based on the graph previously developed, the networkX package was used and the top 5 keys returned.

2 Exercise 2

Regarding exercise 2, we selected the test collection of dataset "500N-KPCrowd". We parsed the xml files and used a TfidVectorizer, with ngrams between 1 and 3 and without stop-words. We parsed the Json file containing the most relevant keyphrases estimated by an expert. As of priors, there were created four different ways: based on the tf-idf scores, based on the bm25 scores, based on a combination between length and position of the candidate in the document and based on a combination between word count and position of the candidate in the document. Regarding these two last priors, they were estimated by giving a bigger weight to candidates that appear in the first sentences, by using the total number of sentences as weights and degrading it while we reach the rest of the sentences (in the case of position). To combine the metrics (length with position and word Count with position), we estimated each metric and then multiplied with the metric in the corresponding combination to obtain the prior values. These values were then normalized. Regarding weights, these were created based on the number of co-occurrences involving the candidates and based on the distributional word similarity between the words in the candidates by using a dataset of pre-trained word embeddings. The priors and the weights were used to estimate the page rank for each candidate and get the top 5 keyphrases. Next, we computed the mean average precision for all the different ways of combining priors and weights on the estimation of the page rank values. The results are in Figure 1. We can see that both the calculations using tf-idf scores were the ones that achieved the best mean average precision. This happened

because these scores were estimated based on the frequency of the terms in the document, so they can achieve best results.

3 Exercise 3

Regarding exercise 3, the dataset used is the same from the previous exercise. This time, we use the documents from the test directory and train directory since we will need data to train the Perceptron for our supervised method and unsupervised method comparison. From these, we extract the keyphrase candidates and their features for each document. The test features extracted are term frequency, inverse document frequency, TF-IDF, BM25 scores, word count and page rank of the candidate. We also extract the correct keyphrases for each document. The train features extracted are the same but with a extra feature to tell if the keyphrase candidate is a keyphrase or not. Now for the Reciprocal Rank Fusion approach, we first rank each feature of each candidate in comparison with the same feature of other candidates in the same document. We use these rankings from the candidate to calculate its Reciprocal Rank Fusion score(RFFscore). After calculating the RFFscores of a document, we select the 5 candidates with the highest score and calculate its precision when compared with the real keyphrases of the document. After doing this for all documents, we calculate the mean average precision of the approach. We compare it with the mean average precision of the supervised method with the Perceptron from the previous part. After testing, we unfortunately obtained a mean average precision of 0 for both methods and we couldn't find any clear fault in our functions. One assumption would be that we lack enough features for our methods.

4 Exercise 4

Regarding exercise 4, we began by parsing any number of given XML/RSS feeds from the New York Times (as long as they are stored on the "links.txt" file) through the use of feedparser, a module for downloading and parsing syndicated feeds.These feeds are stored on a file using the pickle module. Our documents are computed as a combination of the title and description of each feed entry. Like in previous exercises, the candidates were extracted with the TfidfVectorizer with ngrams between 1 and 3 and removing the stop-words. Then, a structure was composed by the term in each phrase resulting in a graph of associations between terms, for every co-occurrence between terms in the same sentence. Based on this graph, the page-rank values were obtained with the networkX module. For presentation, the html is generated to present a word cloud of the top 50 keyphrases ranked, with D3.js, a JavaScript Framework for producing dynamic, interactive data visualizations. This webpage also features a table, related to the top 10 keyphrases, composed of keyphrases, page rank values, number of documents where the keywords are found in the description, and the news articles titles where that specific keyphrase is found in its description.

Mean Average Precision		
Priors/Weights	co-occurrences	similarity
tf-idf scores	0.050196102782818296	0.0500926622581709
bm25 scores	0.02387268728438234	0.022665398740104294
length + position	0.0037419220499715854	0.0035880758961254317
word Count + position	0.001525386987357385	0.001525386987357385

Figure 1: Mean average precision values for different combos of priors and weights used in page rank