Group 19 - Automatic Single-Document Extractive Summarization

Pavlo Kovalchuk
MEIC-A
pavlo.kovalchuk@ist.utl.pt

Gonçalo Fialho
MEIC-A
goncalo.f.pires@ist.utl.pt

Pedro Duarte
MEIC-A
pedro.m.duarte@ist.utl.pt

1. INTRODUCTION

The objective of this project is to create a automatic extractive summarization program by leveraging the PageRank algorithm as well as supervised learning methods. The PageRank method was originally designed to score webpages using it's prestige and importance. In this project, we used this technique on the sentences of some document as a means to rank them by order of importance for an extractive summarization. Note that this project was developed in an Windows environment and folder paths on another OS may need to be changed.

2. EXERCISE 1

2.1 An approach based on graph ranking

In this exercise we implemented a solution for document summarization based on PageRank. This method is based on representing documents as a graph, where nodes correspond to sentences, and the edges encode similarity between sentences.

For simplicity we used the tfIdfVectorizer() from sklearn library to compute the tf-idf weigths for each sentence. Then we compute the cosine similarity of each sentence against the entire colection.

In the next step we start building the graph where the nodes correspond to sentences from the document, and where the edges encode the fact that a pair of sentences has a cosine similarity above the threshold value of 0.2.

Then we rank the sentences according to a variation of the PageRank algorithm, which computes a score for each sentence according to an iterative procedure, where for each iteration we test the convergence of the PageRank values which is set to 0.0001 (based in [2]). In non-converging cases we limit the loop to 50 cicles.

2.1.1 Results

To test the correctness of this approach, we created an example document with some similar phrases and verified if the resulting summary was close to what we were expecting. We did not have any annotated corpus to evaluate with precision, however, we think that the results were within our expectations.

3. EXERCISE 2

3.1 Simple Approach

The simple approach is just an implementation of exercise 1 with the TeMario dataset. The metrics are calculated by comparing the results to the automatic ideal summaries.

3.2 Complex Approach

In order to improve the Page Rank Algorithm now we consider a non-uniform prior probability for each sentence, and also weighted edges in the graph, this technique will provide different outcomes for the ranking of the sentences.

3.2.1 Prior Variants

We implemented different prior variants, listed below:

Sentence Position: Priors are given accordingly the position of the sentence in the document, where the initial sentences will be assigned higher values.

Cosine Similarity: The priors are assigned by the similarity of each sentence agaist the entire document (instead of pairs of sentences).

Title Similarities: Priors given the similarity of the sentence against the title of each document.

Absolute Lenght: We compute the prior by the size of each sentence in the document, where the smallest sentences will have lower priors than the bigger ones.

Relative Lenght: In this case instead of just considering the size of the sentence we also take in consideration the size of entire document.

Graph Centrality: We use the algorithm presented in the lectures where we compute the degree centrality for each node and assign that to the prior.

3.2.2 Weight Variants

For weight variants we use:

Cosine Similarity: Weights are calculated as the similarity between sentences

Noun Phrases: The weights are calculated acording to the number of noun phrases shared between sentences.

Naive Bayes: In this variant we deduced a formula based in the bayes theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where

$$P(A|B)$$

is the probability of a sentence with a given cosine similarity been accepted for the summary

where

$$P(B|A)$$

is the similarity of a given sentence against the entire documente collection

where

$$P(A)$$

is the probability of each sentence in the summary

where

$$P(B)$$

is the probability of a given sentences been accepted in the summary without consider any similarity

3.3 Results

After exhaustive manual research and tests we found the best complex approach. To find it we combined different prior and weight variant and concluded that the best combinaton is Relative Length prior with Naive Bayes weight.

The metrics are calculated below:

Table 1: Exercise 2 Results

Simple approach		Best Complex approach	
Precision	0.394	Precision	0.406
Recall	0.218	Recall	0.223
F1-Score	0.276	F1-Score	0.282
MAP	0.154	MAP	0.160

We concluded that the metrics of complex approach vary a lot according the combination of prior and weight variants. This means that some combinations are even worse than the Simple Approach and it was really interesting to watch the arrangement of them.

4. EXERCISE 3

In this exercise it was asked to use a supervised learning approach to solve the summarization problem, also known as learning-to-rank. The idea is to train a classifier to rank the sentences in a document according to its relevance to a summary. We did this by separating the problem in retrieve & rank problems: the classifier first learns to classify a sentence as relevant or not; then, we rank the initial retrieved sentences by scoring the sentences as a weigthed mean of the page rank and the cosine similarity scores.

4.1 Training the classifier

To train the classifier we used the TeMario 2006 corpus. As a training input we used all the texts in the ”Original” folder and subfolders, and as a target we used all the summarizations in the ”SumariosExtractivos” folder and subfolders. The learning process was done by using the sklearn module.

4.2 Testing the classifier

We tested various classifiers like the Naive Bayes, Perceptron or DecisionTree classifiers and also various features, which will be described in the next subsection. To test the classifier we used all the texts in the ”Textos-fonte” and ”Extractos ideais automaticos” from the original TeMario corpus, and compared the various combinations of classifiers, features and ranking approaches (i.e. use only the position of a sentence instead of the weighted mean of the page rank and cosine similarity as a score). Note that this testing could not be exhaustive due to time restrictions, but we believe we achieved satisfactory results.

Also note that the PCA algorithm could be used to further improve the experimentation process, as it chooses the features that better describe the sentences and therefore reduces the dimensionality of the problem.

4.3 Features

Some features from the following list have been taken from [1].

Occurence of Proper Names: the idea is that a sentence is relevant for a summary if it references someone, some organization, etc. The extraction of this names can be done by leveraging NER (Named Entity Recognition). We used the POS tagger from nltk to extract proper nouns.

Main Concepts: this feature is derived from the fact that sentences that mention a concept that occurs a lot in the rest of the text is also relevant.

Sentence Length: the idea is that a longer sentence is probably more relevant than a short one.

Title Similarity: a sentence that is similar to the title is probably more relevant that a different one. We assume that the first sentence in any text is the title.

Sentence Position: leverages the idea that the first sentences in a text are usually a good summary of the rest of the text.

Cosine Similarity: a sentence is relevant if it is similar to the rest of the document, as discussed in the first project of this course.

Page Rank: similarly, a sentence that has a higher score of page rank has more prestige than others, as discussed in the theoretical classes.

Note that some features of this list are better in the scope of News summarization than normal text summarization, but because News summarization was the main focus of this project this is not an issue.

5. EXERCISE 4

5.1 Extracting RSS feeds and Pre processing

In order to make this exercise work correctly an Internet connection is need as the xml are retrieved from the web. To extract the information uniformly, since their structure is similar, we collect all relevant tagged data: title, description and link.

To prevent xml and html tagging inside the relevant texts we pre process it in order to remove non textual information (i.e. tags). Then we gather all the texts and create a document with all the retrieved news.

5.2 Summarize Approaches

To create the summaries we used the algorithms implemented in previous exercises (1, 2 and 3).

5.3 Generation HTML

After generating the resulting summaries we link them to their source so the html file can redirect the user to the webpage of that specific summary. We use dominate [2] library to create a simple report in html that contains the description of the summary, the journal reference and the link.

To compare the top 5 summaries between them we search for noun phrases (topics) and highlight in the report the top 3. If the top topics are in most of the sentences of the summary it means that the news are highly correlated and the algorithm did a good summarization of the sentences.

The generated html file is named 'summaries.html' and has three tables, one for each exercise.

6. REFERENCES

[1] **N**eto J. , Freitas A. , Kaestner C. , (2002) Automatic Text Summarization using a Machine Learning Approach

[2] **D**ominate , https://github.com/Knio/dominate