Summary Extraction based on Sentence Clustering

Emanuel Felipe Principe de Carvalho
ID:1036121
NLP, SoCS, UoGuelph

April, 2019

Contents

Chapter 1

Introduction

The World Wide Web comprises more than 5.61 billion pages[3]. The amount of information produced and available only tends to increase and filtering these information to suit the needs of one may be challenging. Indexation tools are quite effective for this purpose, but users tend to rely on summaries to refine their option before committing to fully read a piece of text. In the scientific field have been used for that and are an integral part of an article. For the majority of documents, however, that summarization is not available.

Preparing summaries for documents has cost human-associated, sometimes with expert knowledge requirements. Some techniques in Natural Language Processing are directed to approach the complexity of automated text summarization and they are divided in two major groups: Extractive, and Abstract techniques. The former historically uses some linguistics or statistical score to rank sentences and extract the most meaningful ones to build a summary, the latter identifies topics of relevance inside a text and, through Natural Language Generation, creates a summary.

In this work, we explore one extractive alternative of document summarization inspired by the works of Zhang & Li[7], and Jo[6]. The idea is to generate a matrix of similarities between sentences, and then cluster them into topics. The center of each set is then assigned into a summary and compared a human-generated one.

Chapter 2

Methods

The project was developed using Java JDK in the Eclipse Photon environment. The source files are described below as well as the selected database.

2.1 DataSet

The selected dataset was created using BBC news in English from 2004 to 2005 published by Greene and Cunningham in 2006[4] comprised of 2225 articles in five categories: business, entertainment, politics, sports, and tech. The database is available at kaggle and all rights belong to BBC. The data is organized in files, each containing one article and a respective summary made by users through sentence selection. The article was quite impactful, being cited by 166 others.

2.2 Preprocessing

The main class created or the preprocessing was the Scanner. Two support classes are used by it to structure the data. The Scanner receives as arguments the names of the directories for articles and summaries respectively. It then streams the files, creating a new instance of the Article support class for each that contain the sentences and the summary.

Each sentence is created by splitting strings using the characters ".", "!", and "?" as delimiters. The first challenge was here as "." was used in the text for decimal number representation and to finish a sentence. To avoid this problem, if a the final character of a sentence as well as the first one of the subsequent sentence were digits, they were merged back together. The next step was the removal of every word that was not composed solely of letters and had length lesser or equal to 1 and the indexation of each summary sentence.

The final preprocessing phase is the generation of a bibliography file containing the total number of articles followed by their content in the format insequence:

```
$ARTICLE number_of_summary_indexes total_of_sentences
summary_index_1 summary_index_2 ... summary_index_n
sentence_1
sentence_2
⋮
sentence_m
```

2.3 Similiarity Evaluation

The summarization de facto starts here with the options passed to the Summarizer class in no particular order:

- "-k": uses the K estimate for the K-means described by Ramiz[1], more details bellow.
- "-f": save output to a file named scores.txt.
- "-s": output summary lines along with scores.

The Summarizer reads the bibliography file and generate a collection of reports and process them independently. For each report, a Similarity matrix is build where each row and column corresponds to a sentence of the report. This value is given by the formula 2.1 where $tf(d_1, d_2)$ is the number of terms that appear in both sentences and $tf(d)$ is the total of unique terms inside the sentence. This number ranges from 0 to 1 according to how different or similar the sentences are respectively. It is possible to estimate a distance between sentences as the inverse of their similarity.

$$Sim(s_1, s_2) = \frac{tf(d_1, d_2)}{tf(d_1) + tf(d_2)} \tag{2.1}$$

2.4 Clustering and Sentence Selection

The next step was to cluster sentences into sets with similar topics. The method used was the K-means, also implemented for this project. One important aspect is the number k of clusters, by default k equals the number of sentences in the given summaries, but by using the ”-k” option when starting the Summarizer, a estimate proposed by Ramiz[1] can be used. It is described in the equation 2.2, where n is the number of sentences in the article, the denominator is the total of terms in the document and the numerator is the sum of term in each sentence.

$$k = n * \frac{\cup_{i=1}^{n} S_i}{\Sigma_{i=1}^{n} S_i} \tag{2.2}$$

The k-means1 randomly select k sentences as cluster centroids, assign each other sentence to these clusters, and recalculate the centroids, iterating until the sentences are no longer reassigned as shown below. The sentences selected to compose the final summary are the centroids of each cluster. As the K-Means1 is not a deterministic methods, results may vary. No particular order is establish for the summary composition due to time constraints, though.

Algorithm 1 K-Means

```
function K-MEANS(k, Sentences)
    Randomly select k sentences and assign them as Centroids
    repeat
        Assign all sentences to the closest centroid.
        Change the centroid of each cluster to the sentence with lowest accumulative similarity
    until No sentence is reassigned to a different cluster.
end function
```

2.5 Summary sentence selection and evaluation

The implemented metric was the F1 Score and the comparison is made between the indexes predicted by the proposed method and given ones. Given those sets it is easy to find the number of True Positives 2.3a, False Positives 2.3b, False Negatives 2.3c, and consequently calculate the Precision 2.3d, Recall 2.3e, and finally the F1-Score 2.3f.

$$TruePositives = |Labels \cup Prediction| \tag{2.3a}$$

$$FalsePositives = |Prediction| - TruePositives \tag{2.3b}$$

$$FalseNegatives = |Labels| - TruePositives \tag{2.3c}$$

$$Precision = \frac{TruePositives}{TruePrositives + FalsePositives} \tag{2.3d}$$

$$Recall = \frac{TruePositives}{TruePrositives + FalseNegatives} \tag{2.3e}$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.3f}$$

The final output is named "scores.txt" and list the mean for all scores calculated on the first line, followed by the F1-Score for each article individually.

Chapter 3

Results and Discussion

The results for this method and 4 others can be found in the table 3.1. Although the method is quite naive, not approaching dependencies such as proximity between words, it reached a similar result to modern techniques. The k estimation usually would provide two to three times the number of sentences in the original summary and, because of that, was not selected as the default method for calculating the results. The order in which each sentence appear is not important as long as the sentence is in the summary.

3.1 Limitations

Most of the limitations are related to time constraints. The first one that comes arises is the implementation of only one metric for comparison. The main idea of this project was to implement the method proposed by Zhang[7] and improve on it, but the paper lacked information especially on the coefficients of the linear combination of the Word Order Similarity and the definition of the Sim function used for the Word Semantic Similarity. Those alone limited the implementation a lot, but I opted to keep the fundament of clusterization with a different metric described by Jo[6], even though the author built a summary of a set of documents instead.

h!

Method	F1-Score
Proposed Method	0.44174
Zhang[7]	0.47576
MMR[2]	0.43245
WAA[5]	0.45335

Table 3.1: F1-Scores for extractive techniques

Another important delay to the project was the search for a suitable dataset. Most of the summarization databases are better adapted for abstractive techniques and I had to rely on Kaggle to find this one. Many works have referenced this data, 166 cited up to the date that I write this report. The platform also ended up constraining my work, as most of the k-means that have been found rely on absolute spatially positioned instances instead of taking an input of precalculated distances, which speed up a lot the clustering. I ended up implementing all from scratch and not using other tools, but the experience was enlightening.

3.2 Possible Improvements

- ROUGE-N would be a great addition for comparison purposes.
- Other similarity measures for testing purposes.
- Word Embeddings for calculating similarity as inverse of distance.
- Classifiers and potentially Deep Learning methods maybe associated with the previous item would be a great field of exploration.
- Different clusterization methods such as DBScan.
- Implementing a ranking between clusters so that order becomes a factor in the calculations of the results.

Chapter 4

Conclusion

In this work, we developed an extractive summary technique based on pre-calculated similarity measures that were used to cluster sentences into topics and then select the most representative sentence from each topic to compose a final summary. The work a challenge since the initial choice as I was not intimate to most, if not all, of the main fields of study in Natural Language Processing. Even without the experience, I felt happy for building a tool from scratch that, despite its simple idea, achieve results not far from published methods.

Bibliography

[1] Ramiz M. Aliguliyev. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Expert Syst. Appl.*, 36(4):7764–7772, May 2009.

[2] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336, New York, NY, USA, 1998. ACM.

[3] (Maurice de Kunder. The size of the world wide web (the internet), Apr 2019.

[4] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press, 2006.

[5] Wenqian Ji, Zhoujun Li, Wen-Han Chao, and Xiaoming Chen. A new method for calculating similarity between sentences and application on automatic abstracting. *Intelligent Information Management*, 1:36–42, 01 2009.

[6] T. Jo. K nearest neighbor for text summarization using feature similarity. In *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, pages 1–5, Jan 2017.

[7] P. Zhang and C. Li. Automatic text summarization based on sentences clustering and extraction. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 167–170, Aug 2009.