Project 18: Automatic text summarization 1

https://github.com/tHerttua/nlpProject

Jasmin Al Amir Teemu Herttua Armi Korhonen Niina Mäkinen

***Abstract*—Automatic text summarization is a tool to compress the main information of a document to a smaller space. A common approach is to extract the most important sentences from the original text using some kind of scoring algorithm. One of the downsides of extractive methods is the lack of coherence in the produced summary. In this work we use the open-source summarization module PyTLDR and implement an extension, which improves the coherence of outputted text. RESULTS ...**

I. Introduction

With the rapid growth of news articles and other texts on the internet, finding relevant information becomes more and more laborious. Summarization of documents for faster overviews has a growing demand to be able to spend less time reading and searching for suited articles. Manually summarizing documents is slow and inefficient, which calls for tools for automatizing the task.

Automatic text summarization (ATS) is used to compress an input document to a shorter version which still contains the relevant information of the original text. Luhn [1] introduced the concept in 1958. He used word frequency and distribution to derive measures of significance for words and sentences. In 1969, Edmundson [2] combined the frequency approach with cue words, and also considered title and heading words and sentence location. It was concluded that using only statistical information is insufficient; semantic characteristics of language need to be taken account of.

More advanced techniques where e.g. relations between sentences and are considered were later developed, and machine learning approaches started gaining interest when the amount of available training data increased [3]. Current catalog of methods is extensive as the field has become more popular. A summary of the pros and cons of the current methods can be found in Tables 1 and 2 of El-Kassas et al. [4]. Many ATS systems utilize multiple methods, which allows the use of their respective strengths and eliminates their shortcomings.

Two approaches for ATS exist: extractive and abstractive. In extractive methods, the most important sentences in the input text are used for the summary. Abstractive methods generate the summary using words and sentences which are not in the original document. Hybrid approaches have also been used [4]. Since the abstractive approach is more complex, most of the research has been focusing on extractive methods.

The basic workflow of extractive summarization starts with forming a suitable representation of the input text. This is dependent of the extraction method, and can mean e.g. a graph or a matrix representation of the words and sentences. The sentences are then scored using some algorithm, and the sentences with the highest scores are extracted. The problem is then to form the algorithm, and to design the representation type. Some post-processing of the sentences can optionally be done after extraction.

ATS is a challenging task, and current methods are not able to produce summaries similar to those written by humans [4]. Extractive techniques can produce incoherent summaries, where sentences seem disconnected. Using natural language processing is needed for producing more human-like summaries.

Coherence of text can be enhanced e.g. with respect to named entity. It refers to an abstract or physical object, which can be assigned a name. Recognizing named entities from documents is a challenging task, mostly approached using supervised machine learning techniques [5]. Several open-access tools have been developed for the task.

Developing of automated systems is dependent of an appropriate evaluation metric. Standard evaluation techniques used in ATS compare automatically and manually produced summaries and describe their similarity. While this is a valid approach, there is not necessarily one single way of making a good summary. The evaluation of the goodness of a summary can be subjective, and finding a functioning metric can be challenging. A common choice is to use the ROUGE framework [6], which has been shown to correlate well with the opinions of human evaluators [7].

PyTLDR[1] is an open-source Python module used for extractive document summarization. It includes three implementations of ATS: a TextRank based, a Latent Semantic Analysis based and a sentence relevance score based summarizer. In this work we evaluate the summarization algorithms provided by the PyTLDR module. We use the CNN/Dailymail[2] data set and the standard ROUGE-N evaluation metric to asses the performance of the methods. We propose a new algorithm for increasing coherence of text with respect to named-entity using the recognition framework in the spaCy[3] module. The algorithm is integrated with the original PyTLDR implementations and their updated performance is evaluated using ROUGE-N. We develop a Graphical User Interface (GUI) for easy

[1]https://github.com/jaijuneja/PyTLDR
[2]https://github.com/morningmoni/FAR
[3]https://github.com/explosion/spaCy

utilization of the software.

This report is organized as follows: Section II presents the theoretical foundation of the work. The software and workflow are described in Section III. The results are presented Section IV and further discussed in Section V. Section VI contains the concluding remarks.

II. Background

A. TextRank

TextRank [8] is a graph-based ranking model for text processing. A generalization of PageRank, used to score sentences based on their relevance. It is fully unsupervized and does not require training corpora.

The idea behind graph-based ranking is to decide the importance of a vertex in a graph based on the votes cast by other vertices. The votes are links between vertices, and the more links to a vertex are cast, the more important it is. The score S of a vertex V_i is defined as [9]

$$S(V_i) = (1-d) + d \sum_{j \in In(V_i)} \frac{S(V_j)}{|Out(V_j)|}, \tag{1}$$

where $0 < d < 1$ is a damping factor, $In(V_i)$ denotes the set of vertices pointing to V_i and $Out(V_j)$ denotes the set of vertices V_j points to. In TextRank, the strength of the connection between vertices V_i and V_j is included by using edge weights w_{ji}. Weighted score $WS(V_i)$ is defined as

$$WS(V_i) = (1-d) + d \sum_{V_j \in In(V_i)} \frac{w_{ji} WS(V_j)}{\sum\limits_{V_k \in Out(V_j)} w_{jk}}. \tag{2}$$

The TextRank algorithm is iterative, starting with random values assigned to the vertices and iterating until convergence.

The vertices in a graph are complete sentences, and the highest-scoring ones are used in the summary. The vote casting between sentences is based on their similarity, e.g. word overlap. The overlap-based similarity of two sentences S_i and S_j, defined as sets of words w, is [8]

$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)}. \tag{3}$$

The numerator is a normalization factor ensuring that long sentences are not favored. Other similarity methods can also be used.

The main steps of implementing graph-based algorithms in natural language processing are (from Mihalcea and Tarau, 2004 [8])

1) Identify text units that best define the task at hand, and add them as vertices in the graph.
2) Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
3) Iterate the graph-based ranking algorithm until convergence.
4) Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

B. Latent Semantic Analysis

Latent Semantic Analysis applies Singular Value Decomposition (SVD) to the text summarization task (e.g. [10], [11], [12], [13]). The SVD of a $n \times m$ matrix $\mathbf{A}$ is defined as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{4}$$

where $\mathbf{U}$ is a $m \times m$ unitary matrix, $\mathbf{\Sigma}$ is a $m \times n$ matrix and $\mathbf{V}$ is a $n \times n$ unitary matrix. The elements of $\mathbf{\Sigma}$, placed on the diagonal, are called singular values of $\mathbf{A}$. The columns of $\mathbf{U}$ and $\mathbf{V}$ are the left- and right-singular vectors of $\mathbf{A}$. The SVD provides a way to express $\mathbf{A}$ as a sum of its constituents:

$$A = \sum_{k=1}^{l} \sigma_k u_k v_k^T, \tag{5}$$

where $l = min\{n, m\}$, σ_k is the kth singular value and u_k and v_k are kth columns of $\mathbf{U}$ and $\mathbf{V}$, respectively. The magnitude of each singular value indicates the relevance of each part of the sum, and since they are ordered, SVD can be used to obtain the components in order of importance.

In case of text analyzing, the matrix $\mathbf{A}$ is formed such that columns represent sentences and rows are words or phrases. The matrix entries can be e.g. frequency of word in corresponding sentence or Tf-idf [11]. SVD captures patterns, and singular vectors can then be thought as concepts in the text and singular values as the relevance of each concept [13].

Gong et al. [13] suggested that the rows of the matrix $\mathbf{V}^T$ consists of these concepts and the columns are sentences. Each matrix entry denotes an index, which indicates how well the sentence describes the corresponding concept. In their ATS approach, a predefined number of important concepts was chosen. Then, the sentence with the highest index with respect to each concept was extracted for the summary.

Steinberger et al. [12] pointed out that the above method leaves out relevant sentences, which have a high but not the highest index. They also noted that the number of sentences one wants to include in the summary is the same as the number of extracted concepts, and the relevance of the concepts decreases with the number of sentences. They proposed a modified method, where a vector $\mathbf{s}$ with components

$$s_k = \sqrt{\sum_{i=1}^{r} \sigma_i^2 v_{k,i}^2}, \tag{6}$$

where r is the dimension of the new space and v_k is a sentence vector in $\mathbf{V}$, is calculated. In essence, sentence vectors are weighted with their corresponding singular value, and their length is calculated. The sentences with the highest s_k are extracted for the summary. The parameter r is set such that only the dimensions whose singular value is over some predefined threshold, usually $0.5\sigma_{max}$, are included.

Sentences can belong to more than one main concept, which causes noise to the $\mathbf{V}^T$ matrix. Ozsoy et al. [10] proposed the Cross method as an extension to the approach of Steinberger et al. to overcome this issue. They added a pre-processing step for the $\mathbf{V}^T$ matrix, in which the average sentence score is calculated for each topic. The cells with an index value equal or less than the average score are set to zero. The algorithm then proceeds essentially identically to the steps in Steinberger et al.

C. Relevance sentence scoring

A simple method for extracting sentences is to score them by relevance, and then choose the highest scoring ones. In the relevance scoring algorithm described by Gong et al. [13], the document is decomposed into sentences. A weighted term-frequency vector is created for the document and for each individual sentence, and the inner products of the sentences with the whole document are calculated. The highest scoring sentence is extracted for the summary, and the terms it contains are removed from the document. The procedure is repeated until the pre-defined number of sentences is attained. The workflow is as follows (from Gong et al. [13]):

1) Decompose the document into sentences and form the candidate sentence set S.
2) Create weighted term-frequency vectorA_i for each sentence in S and a weighted term-frequency vector D for the whole document.
3) Compute the inner product between D and each A_i.
4) Add the highest scoring sentence to the summary.
5) Delete the highest scoring sentence from S, and eliminate all the terms in the sentence from the document. Recompute D.
6) If the number of sentences in the summary has reached the pre-defined value, terminate. Otherwise go to Step 3.

D. ROUGE-N

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [6] is the most widely used evaluation tool in the field of ATS [4]. It is used to determine the quality of an automatically produced summary by comparing it to manually produced, ”ideal” versions.

Several types of ROUGE measures exist. We use ROUGE-N, which is defined as the n-gram (sequence of n adjacent words in the text) recall. ROUGE-N recall R between the generated and one reference summary is calculated as

$$R = \frac{N_{overlap}}{N_{reference}}, \tag{7}$$

where $N_{overlap}$ is the number of overlapping n-grams in the evaluated and reference summaries, and $N_{reference}$ is the number of n-grams in the reference summary. It describes how much of the reference summary is captured in the automatically produced version.

The downside of recall is that summaries can achieve higher scores by simply increasing the length. This can lead to redundant words and sentences, which is counterproductive to the purpose of summarizing. It is then necessary to include precision P of the summary

$$P = \frac{N_{overlap}}{N_{generated}}, \tag{8}$$

where $N_{generated}$ is the number of n-grams in the generated summary. Precision and recall can be expressed jointly using the traditional F-score

$$F = \frac{2PR}{P + R}, \tag{9}$$

which is the harmonic mean of the two.

III. Workflow

A. GUI

The user can select from three summarization methods: TextRank, LSA and Relevance. The LSA approach is implemented using the Cross method by Oszoy et al.

NEED:
- Print screen from the GUI view
- Some pipeline graph (input - processes - output)

B. Data sets

The CNN/Dailymail used in Mao et al. [14].

IV. Results

e.g. average ROUGE-scores per method, possibly with min+max included if large variance varies (will show consistency/lack of).

A. Original PyTLDR

(Task one was to test out with own text. It seems out of place, so I wonder if it is really needed.)

B. ROUGE-N scores of CNN/Dailymail data set

C. ROUGE-N scores of CNN/Dailymail using named-entity heuristic

(Bad section titles, will need to figure out how to represent)

V. Discussion

The automatic generation of precise, relevant and coherent summaries is a prominent problem in natural language processing. Methods for the task have been developed since the 1950s.

Due to the explosive growth of information on the internet, the need for such systems is continuously growing.

Extractive techniques are the most popular choice.

Coherence ...

We have developed a software for automatically summarizing ...

The techniques we are using ...

Our results ...

Possible follow-up ...

VI. Conclusions

We have developed a software which does a very good job.

References

[1] H. P. Luhn, ”The Automatic Creation of Literature Abstracts,” in IBM Journal of Research and Development, vol. 2, no. 2, pp. 159-165, Apr. 1958

[2] H. P. Edmundson, ”New Methods in Automatic Extracting,” Journal of the ACM, vol. 16, no. 2 oo. 264-285, Apr. 1969

[3] G. Erkan, D. R. Radev, ”Lexrank: Graph-based lexical centrality as salience in text summarization,” Journal of artificial intelligence research, vol. 22, pp. 457-479, Dec. 2004

[4] W. S. El-Kassas, C. R. Salama, A. A. Rafea, H. K. Mohamed, ”Automatic Text Summarization: A Comprehensive Survey,” Expert Systems with Applications, vol. 165, 113679, Mar. 2020

[5] R. Jiang, R. E. Banchs, and H. Li, ”Evaluating and combining name entity recognition systems,” In Proceedings of the Sixth Named Entity Workshop, pp. 21-27, Aug. 2016

[6] C. Y. Lin, ”Rouge: A package for automatic evaluation of summaries,” in Text summarization branches out, pp. 74-81, Jul. 2004

[7] R. Sun, Z. Wang, Y. Ren, and D. Ji, ”Query-Biased Multi-document Abstractive Summarization via Submodular Maximization Using Event Guidance,” International Conference on Web-Age Information Management, 2016, pp. 310–322

[8] R. Mihalcea, P. Tarau, ”Textrank: Bringing order into text,” in Proceedings of the 2004 conference on empirical methods in natural language processing, pp. 404-411, Jul. 2004

[9] S. Brin and L. Page, ”The anatomy of a large-scale hypertextual Web search engine,” Computer Networks and ISDN Systems, vol. 30, pp. 107-117, 1998

[10] M. Ozsoy, I. Cicekli, and F. Alpaslan, ”Text summarization of turkish texts using latent semantic analysis,” In Proceedings of the 23rd international conference on computational linguistics (Coling 2010), pp. 869-876, Aug. 2010

[11] M.G. Ozsoy, F. N. Alpaslan, I. Cicekli, ”Text summarization using Latent Semantic Analysis,” in Journal of Information Science, vol. 37, no. 4, pp. 405-417, Jun. 2011

[12] J. Steinberger, K. Jezek, ”Using latent semantic analysis in text summarization and summary evaluation,” Proc. ISIM, vol. 4, pp. 93-100, 2004

[13] Y. Gong, X. Liu, ”Generic text summarization using relevance measure and latent semantic analysis,” in Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 19-25, Sept. 2001

[14] Y. Mao, L. Liu, Q. Zhu, X. Ren, and J. Han, ”Facet-Aware Evaluation for Extractive Text Summarization,” arXiv, arXiv-1908, Aug. 2019