Unsupervised document summarization using pre-trained sentence embeddings and graph centrality

Workshop on Scholarly Document Processing 2021 LongSumm shared task

Juan Ramirez and Evangelos Milios

Dalhousie University

Introduction

The essential idea of this work is that, while the sentence embeddings produced by SBERT are not well suited for clustering algorithms like Hierarchical Clustering or DBSCAN, they produce excellent results in Paraphrase Identification or Semantic Textual Similarity when compared with Cosine Similarity, which implies that they can be used along with graph centrality methods. The text summarization method proposed in this paper has the following contributions:

- Is unsupervised and can be used as a proxy for more advanced summarization methods.
- Can easily scale to arbitrarily large amounts of text.
- Is fast and easy to implement.
- Can fit any length requirements for the production of summaries.

Methodology

The system is composed of three main steps: first, we use SBERT to produce sentence embeddings for every sentence in the document to summarize; next, we form a graph by comparing all the pairs of sentence embeddings obtained and finally, we rank the sentences by their degree centrality in this graph. Fig. 1 gives an overview of the whole method.

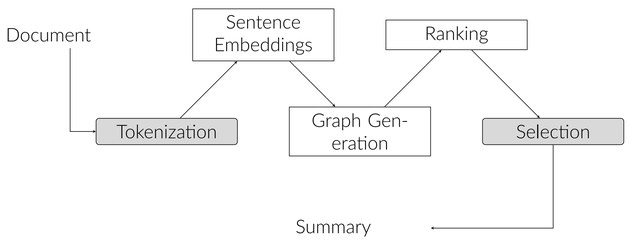

Figure 1:The complete pipeline of the proposed method. In the first step, we split the input text into sentences by using a regular expression handcrafted specifically for scientific documents. In the second step, we compute the sentence embeddings of the parsed sentences using SBERT. In the third step, we create a graph by comparing all the pairs of sentence embeddings obtained using cosine similarity. In the fourth step, we rank the sentences by the degree centrality in the generated graph. In the fifth and final step, we only keep a certain number of sentences or words to adjust to the length requirements of the summary.

Sentence tokenization

The first step of our pipeline is to split the input text into a list of sentences. This step is critical because if the sentences are too long, the final summary will have a lot of meaningless content (therefore losing precision). However, if the sentences are too short, there is a risk of not having enough context to produce an accurate sentence embedding for them or extracting meaningless sequences, like data in tables or numbers that lie in the middle of the text. We found that the function $\mathrm{sent\_tokenize()}$ from the NLTK package often failed because of the numbers in the tables and the abbreviations, like "et al.", which are very common in scientific literature. Because of this, we used a regular expression handcrafted specifically to split the text found in scientific documents.

Computing sentence embeddings

After extracting the sentences, the next step is to produce the sentence embedding of each sentence using SBERT , which is a Transformer-based model built on top of BERT that takes as input sentences and produces sentence embeddings that can be compared with cosine similarity, which is given by the following formula:

$$sim(x, y) = \frac{x \cdot y}{|x||y|}.$$

These sentence embeddings are superior in quality than taking the CLS token of BERT or averaging the sentence embeddings of the words in the sentence produced by BERT, GloVe , or Word2Vec .

SBERT, like BERT, was pre-trained on a general large text collection to learn good sentence embeddings, but it has to be fine-tuned on a more specific data set according to the task. Since we are working with scientific papers, we picked the "base" version of RoBERTa that was fine-tuned in the MSMARCO data set for the Information Retrieval task.

Generation of the sentence graph

After the sentence embeddings have been produced, the next step is to produce a weighted complete graph with a node for each sentence in the text. Its edges are weighted according to the cosine similarities of the corresponding sentence embeddings. An example graph is depicted in Fig. 2.

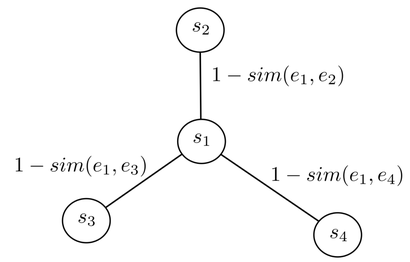

Figure 2:The process of graph generation and ranking of the sentences. Every node in the generated complete graph represents a sentence in the document and the weight of each edge is given by the similarity between the nodes it conects. The importance of the sentence in the document is modelled as $rank(s_i) = \sum_{j=1}^{n} 1 - sim(e_i, e_j)$, where e_i and e_j are the corresponding SBERT sentence embeddings of s_i and s_j.

Ranking by centrality

The forth step is to assign a score for each sentence that allows us to sort them by their importance in the document. As a consequence, we define the importance rank for each sentence as follows:

$$rank(s_i) = \sum_{j=1}^{n} A[i, j] = \sum_{j=1}^{n} 1 - sim(e_i, e_j),$$

where e_i and e_j are the corresponding SBERT sentence embedding for s_i and s_j.

Summary selection

The final step in the method is to select the sentences that are going to form the summary. To do this, we can take only the bottom n-percentile in reverse (as opposed to the top n-percentile, since in our method, a lower rank means that the sentence is more important in the document) or concatenate the ranked sentences in reverse (so that the sentences with the lowest ranks -that is, the most important ones- come first) and take the first k words to satisfy a word-length constraint for the summaries.

Results

Overall, we observed that the 600-word constraint of the task prevented our method from performing better, but we also observed that the best summaries produced by our method are too long (around 1,000 words or more). Table 1 displays the performance of the method variations that we submitted to the task.

Bottom %	ROUGE-1		ROUGE-2		ROUGE-L		Avg. Length
	F-measure	Recall	F-measure	Recall	F-measure	Recall	
1.0	0.24	0.15	0.06	0.03	0.11	0.07	183.2
1.5	0.29	0.21	0.08	0.05	0.13	0.09	257.0
2.0	0.33	0.25	0.08	0.06	0.14	0.10	314.8
2.5	0.37	0.29	0.09	0.07	0.15	0.11	366.7
5.0	0.44	0.39	0.12	0.10	0.16	0.14	530.5
10.0	0.46	0.43	0.12	0.12	0.17	0.16	591.3
15.0	0.46	0.43	0.12	0.12	0.17	0.16	597.0

Table 1: Performance of the different variations of the proposed method submitted to the task. In this setting, the ranked sentences were sorted in reverse and concatenated to form a preliminary output, which was truncated at 600 words to comply with the task's requirements. The "Bottom %" column displays the percentile used in the sentence selection phase of the method. Avg. Length displays the average length in words of the summaries produced for the test set.

Conclusion and Future Work

The method introduced in this work displays competitive performance with more sophisticated methods and can be useful when there is not enough labelled data to train a deep neural summarization system while being fast, simple and efficient. Overall, we observed that the recall component of ROUGE for the proposed method has much room for improvement, as having sentences as the minimal text units makes it harder to include relevant phrases that are joined with others that are not so relevant. Another important future direction is to reduce the redundancy of the summaries, as it is common to have several versions of the same important sentence scattered across the document, so all these versions of the sentence appear in the final summary.

Contact information

- email: juan.ramirez@dal.ca
- website: https://auto-summ.herokuapp.com/

DALHOUSIE UNIVERSITY