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

Juan Ramirez-Orta* and Evangelos Milios
Department of Computer Science
Dalhousie University

Abstract

This paper describes our submission for the LongSumm task in SDP 2021. We propose a method for incorporating sentence embeddings produced by deep language models into extractive summarization techniques based on graph centrality in an unsupervised manner. The proposed method is simple, fast, can summarize any document of any size and can satisfy any length constraints for the summaries produced. The method offers competitive performance to more sophisticated supervised methods and can serve as a proxy for abstractive summarization techniques.

1 Introduction

Automatic text summarization is a very old and important task in Natural Language Processing (NLP) that has received continued attention since the creation of the field in the late 50's (Luhn, 1958), mainly because of the ever-increasing size of collections of text. The objective of the task is, given a document, to produce a shorter text with maximum information content, fluency and coherence. The summarization task can be classified into extractive and abstractive. Extractive summarization means that the summary is composed exclusively of passages present in the original document and abstractive summarization means that there can be words in the summary that did not appear in the original document.

Since the creation of the first neural language models (Bengio et al., 2003), vector representations of text that encode meaning (called embeddings) have played a significant role in NLP. They allow the application of statistical and geometrical methods to words, sentences and documents ((Pennington et al., 2014), (Mikolov et al., 2013), (Reimers and Gurevych, 2019)), leading to state-of-the-art performance on several NLP tasks like Information Retrieval, Question Answering or Paraphrase Identification. Among these neural language models, very deep pre-trained neural language models, like BERT (Devlin et al., 2018), T5 (Raffel et al., 2020), and GPT-3 (Brown et al., 2020) have shown impressive performance in tasks like language modelling and text generation or benchmarks like GLUE (Wang et al., 2018).

An important variation of extractive summarization that goes back as far as the late 90's (Salton et al., 1994, 1997) utilizes graphs, where the nodes represent text units and the links represent some measure of semantic similarity. These early graph-based summarization techniques involved creating a graph where the nodes were the sentences or paragraphs of a document and two nodes were connected if the corresponding text units had a similar vocabulary. After creating the document graph, the system created a summary by starting at the first paragraph and following random walks defined by different algorithms that tried to cover as much of the graph as possible.

A more evolved approach was the creation of *lexical centrality* (Erkan and Radev, 2004) (Mihalcea and Tarau, 2004) (Wolf and Gibson, 2004), which is a measure of the importance of a passage in a text where the sentences of the document are connected by the similarity of their vocabularies.

The current state of the art in automatic summarization with graphs is mainly based on algorithms like PageRank (Brin and Page, 1998) enhanced with statistical information of the terms in the document (like in (Ramesh et al., 2014)) or Graph Neural Networks (Kipf and Welling, 2016) on top of deep language models (like in (Xu et al., 2019)).

Only two systems from the previous Scholarly Document Processing workshop held in 2020 are based on graphs: CIST-BUPT and Monash-Summ.

In CIST-BUPT (Li et al., 2020), they used Recurrent Neural Networks to create sentence embeddings that can be used to build a graph which

Please send correspondence to juan.ramirez.orta@dal.ca

Proceedings of the Second Workshop on Scholarly Document Processing, pages 110–115
June 10, 2021. ©2021 Association for Computational Linguistics

is then fed into a Graph Convolutional Network (Kipf and Welling, 2016) and a Graph Attention Network (Veličković et al., 2018) to create extractive summaries. To generate abstractive summaries, they used the gap-sentence method of (Zhang et al., 2019) to fine-tune T5 (Raffel et al., 2020).

In Monash-Summ (Ju et al., 2020), they propose an unsupervised approach that leverages linguistic knowledge to construct a sentence graph like in SummPip (Zhao et al., 2020). The graph nodes, which represent sentences, are further clustered to control the summary length, while the final abstractive summary is created from the key phrases and discourse from each cluster.

This work focuses on extractive summarization using graphs leveraging sentence embeddings produced by pre-trained language models. The essential idea is that, while the sentence embeddings produced by SBERT (Reimers and Gurevych, 2019) are not well suited for clustering algorithms like Hierarchical Clustering or DBSCAN (Ester et al., 1996), 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.

2 Methodology

In this section, we describe how the system works. 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.

2.1 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 `sent_tokenize()` from the NLTK package (Bird et al., 2009) 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.

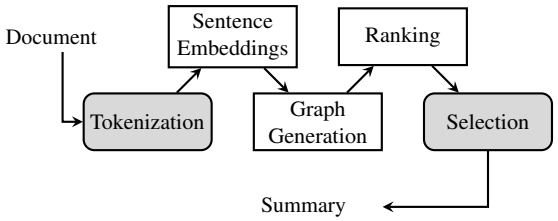

Figure 1: The complete pipeline of the proposed method. In the first step, we split the input text into sentences by using a regular expression handcrafted specifically for scientific documents. In the second step, we compute the sentence embeddings of the parsed sentences using SBERT. In the third step, 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.

2.2 Computing sentence embeddings

After extracting the sentences, the next step is to produce the sentence embedding of each sentence using SBERT (Reimers and Gurevych, 2019), which is a Transformer-based (Vaswani et al., 2017) model built on top of BERT (Devlin et al., 2018) that takes as input sentences and produces sentence embeddings that can be compared with cosine similarity, which is given by the following formula:

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

As shown in (Reimers and Gurevych, 2019), 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 (Pennington et al., 2014), or Word2Vec (Mikolov et al., 2013).

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 (Liu et al., 2019) that was fine-tuned in the MSMARCO data set (Bajaj et al., 2016) for the Information Retrieval task.

2.3 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.

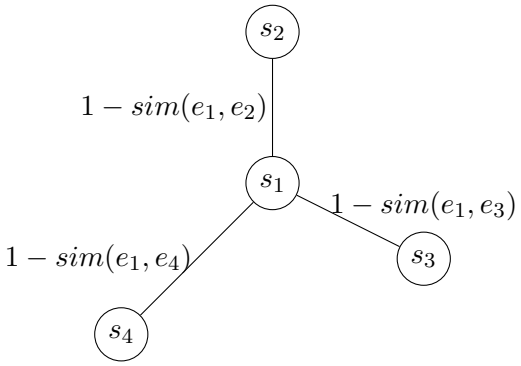

Figure 2: The process of graph generation and ranking of the sentences. Every node in the generated complete graph represents a sentence in the document and the weight of each edge is given by the similarity between the nodes it conects. The importance of the sentence in the document is modelled as $rank(s_i)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

To build this graph, the first step is to gather all the pairwise cosine similarities in a matrix. Let $D = (s_1, s_2, ..., s_n)$ be a document. Using SBERT, we produce a sequence of vectors $(e_1, e_2, ..., e_n)$, where e_i is the sentence embedding of s_i. Then, we can compute the matrix A, where $A[i, j] = 1 - sim(e_i, e_j)$.

We make the following observations:

- The diagonal of A is composed exclusively of zeros, because $A[i, i] = 1 - sim(e_i, e_i) = 0$.
- The matrix A is symmetric, because $A[i, j] = 1 - sim(e_i, e_j) = 1 - sim(e_j, e_i) = A[j, i]$.
- All the entries in A are non-negative, because $-1 \leq sim(e_i, e_j) \leq 1$.

These observations imply that the matrix A can be interpreted as the adjacency matrix of a weighted complete graph $G = (V, E)$ where $V = \{s_1, s_2, ..., s_n\}$, $E = \{(s_1, s_2) | s_1, s_2 \in V\}$ and the edges are weighted by the following function: $w(s_1, s_2) = 1 - sim(e_1, e_2)$.

2.4 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:

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

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

To motivate this definition, we observe that adding the entries of the matrix A columnwise gives naturally a ranking of the nodes of G that is a natural generalization of the degree centrality. However, in our ranking, the most "central" sentences (sentences that are similar to many other sentences in the document) have lower scores than the ones that are less "central."

To further support this definition, we observe that if G were an undirected, unweighted simple graph $G = (V, E)$ (that is, the entries of A are either 0 or 1, A is symmetric and only has zeros in its diagonal), then we would have that

$$\sum_{j=1}^{n} A[i, j] = \#\{v \in V | (v_i, v) \in E\}, \quad (2)$$

which is the definition of the degree of node v_i and is clearly a (somewhat crude) measure of the importance of the node in the graph.

It is important to note that in scientific papers, which have around 300 sentences, the proposed method takes around 1 second for the whole process. This result implies that there is no obstacle for applying this method to longer documents since producing the sentence embeddings with the SBERT implementation is very efficient, and the only thing that we are doing is compare all the pairs of sentence embeddings, which can be done with highly efficient linear algebra libraries.

2.5 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.

3 Experimental setup

3.1 Data set

Since our method is for unsupervised extractive summarization, we only used the extractive summaries in the TalkSumm data set (Lev et al., 2019) to estimate the appropriate threshold value for the sentence selection phase. As suggested in the task, we used science-parse (AllenAI, 2019) to extract the text of the scientific articles and split it into sections. Given that the objective of the task is to produce long summaries for the documents, we discarded the title and abstract and then took as input for the algorithm the remaining text as a single block.

3.2 Evaluation

As is customary in summarization tasks, we used ROUGE (Lin, 2004) in its variations ROUGE-1, ROUGE-2 and ROUGE-L.

3.3 Percentile threshold in the selection phase

We tried with $p = \{1, 1.5, 2, 2.5, 5, 10, 15\}$ as the value of the bottom percentage of sentences to keep for the final summary and truncated the output to satisfy the 600 word limit for the task when the summary was longer. It is important to note that the freedom of this parameter allows the system to produce summaries of arbitrary length, depending on the task at hand.

4 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 %	R-1 F	R-1 R	R-2 F	R-2 R	R-L F	R-L R
1.0	0.24	0.15	0.06	0.03	0.11	0.07
1.5	0.29	0.21	0.08	0.05	0.13	0.09
2.0	0.33	0.25	0.08	0.06	0.14	0.10
2.5	0.37	0.29	0.09	0.07	0.15	0.11
5.0	0.44	0.39	0.12	0.10	0.16	0.14
10.0	0.46	0.43	0.12	0.12	0.17	0.16
15.0	0.46	0.43	0.12	0.12	0.17	0.16

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. R-N F stands for the F-measure in ROUGE-N, while R-N R stands for the Recall in ROUGE-N.

5 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 precision component of ROUGE for the proposed method has much room for improvement, as having sentences as the minimal text units prevents it from filtering out the less important phrases. 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.

References

AllenAI. 2019. Science parse. GitHub repository, https://github.com/allenai/science-parse. Visited on April 23, 2021.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 107–117.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

Jiaxin Ju, Ming Liu, Longxiang Gao, and Shirui Pan. 2020. Monash-summ@LongSumm 20 SciSummPip: An unsupervised scientific paper summarization pipeline. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 318–327, Online. Association for Computational Linguistics.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Guy Lev, Michal Shmueli-Scheuer, Jonathan Herzig, Achiya Jerbi, and David Konopnicki. 2019. TalkSumm: A dataset and scalable annotation method for scientific paper summarization based on conference talks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2125–2131, Florence, Italy. Association for Computational Linguistics.

Lei Li, Yang Xie, Wei Liu, Yinan Liu, Yafei Jiang, Siya Qi, and Xingyuan Li. 2020. CIST@CL-SciSumm 2020, LongSumm 2020: Automatic scientific document summarization. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 225–234, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Animesh Ramesh, K. Srinivasa, and Pramod .N. 2014. Sentencerank — a graph based approach to summarize text. In *5th International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2014*, pages 177–182.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Gerard Salton, James Allan, Chris Buckley, and Amit Singhal. 1994. Automatic analysis, theme generation, and summarization of machine-readable texts. *Science*, 264(5164):1421–1426.

Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Automatic text structuring and summarization. *Information Processing & Management*, 33(2):193–207. Methods and Tools for the Automatic Construction of Hypertext.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International*

Conference on Learning Representations. Accepted as poster.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Florian Wolf and Edward Gibson. 2004. Paragraph-, word-, and coherence-based approaches to sentence ranking: A comparison of algorithm and human performance. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 383–390, Barcelona, Spain.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Discourse-aware neural extractive model for text summarization. *CoRR*, abs/1910.14142.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*.

Jinming Zhao, Ming Liu, Longxiang Gao, Yuan Jin, Lan Du, He Zhao, He Zhang, and Gholamreza Haffari. 2020. Summpip: Unsupervised multi-document summarization with sentence graph compression. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1949–1952, New York, NY, USA. Association for Computing Machinery.