



Interactive Document Summarization

Raoufdine Said, Adrien Guille

► To cite this version:

Raoufdine Said, Adrien Guille. Interactive Document Summarization. 46th European Conference on Information Retrieval (ECIR 2024), Mar 2024, Glasgow, United Kingdom. pp.177-181, 10.1007/978-3-031-56069-9_14 . hal-04521175

HAL Id: hal-04521175

<https://hal.science/hal-04521175>

Submitted on 29 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Interactive Document Summarization

Raoufdine Said, Adrien Guille^[0000–0002–1274–6040]

Université de Lyon, Lyon 2, ERIC UR 3083, France

`raoufdine.said@univ-lyon2.fr`

`adrien.guille@univ-lyon2.fr`

Abstract. With the advent of modern chatbots, automatic summarization is becoming common practice to quicken access to information. However the summaries they generate can be biased, unhelpful or untruthful. Hence, in sensitive scenarios, extractive summarization remains a more reliable approach. In this paper we present an original extractive method combining a GNN-based encoder and a RNN-based decoder, coupled with a user-friendly interface that allows for interactive summarization.

Keywords: Interactive Summarization · Extractive Summarization · Graph Neural Network · Recurrent Neural Network · User Interface.

1 Introduction

One of the most common uses for modern chatbots is automatic summarization [1]. While the abstractive summaries generated by these tools can be relevant, the language models they’re based on might be biased, *e.g.* politically biased [9] or gender biased [5], they can sometimes be untruthful because large language models are prone to hallucinate [6], or they can simply be unhelpful, as chatbots might fail to follow instructions correctly [8]. This limits their reliability and can make them ill-suited to summarize sensitive documents, like scientific articles, encyclopedic articles or press articles, among others, where information shouldn’t be altered. A viable alternative under this scenario is extractive summarization, which consists in composing the summary from actual sentences picked from the original document. In this paper, we first present an original method for extractive summarization, by combining two existing methods. The implementation of the model is available here: <https://github.com/baragouine/radsum>. Next, we present an interface that allows users to interactively summarize documents, the code of which is available here: https://github.com/baragouine/radsum_app/. For a video demonstration, see <https://youtu.be/vBenEaCIwkl>.

2 Summarization Method

We combine two existing methods, namely (i) HeterSUMGraph [11], which we slightly modify and only use for the encoding part of our architecture, and (ii) SummaRuNNer [7], which we use for the decoding part only. Fig. 1 illustrates the overall methodology. The rationale behind combining these methods is to have a more expressive encoder while having an interpretable decoder.

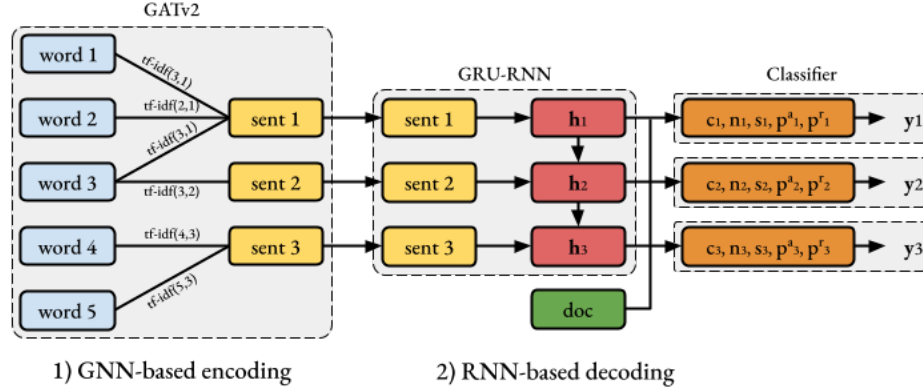


Fig. 1. Overall architecture of the implemented summarization method.

Document Encoding. We convert the input document to a graph as in HeterSUMGraph [11]. It is a bipartite graph with two kinds of vertices: word vertices and sentence vertices. More precisely there is one vertex per unique word in the document and one vertex per sentence in the document, connected according to the composition of the sentences. Edges are weighted with $tf-idf$ scores measured at the sentence level, meaning tf is calculated as the number of times the word occurs in the sentence, while idf is defined as the inverse of the degree of the word vertex. Word embeddings are propagated with a 2-layer graph neural network to obtain sentence representations. Whereas HeterSUMGraph uses GAT layers [10], we implement more expressive GATv2 layers [2], with an attention mechanism that accounts for discretized $tf-idf$ weights via edge embeddings.

Summary Decoding. We pass the sentence representations to a recurrent neural network based on the GRU cell [3] to further contextualize them and proceed similarly to SummaRuNNer in classifying sentences sequentially, following their order in the document. The probability to keep the i^{th} sentence is calculated in terms of 5 scores:

- **content score:** a linear function of the representation of this sentence, $\mathbf{W}_c \mathbf{h}_i$;
- **salience score:** a bilinear function of the representation of this sentence and the representation of the whole document, $\mathbf{h}_i^\top \mathbf{W}_s \mathbf{d}$;
- **novelty score:** a bilinear function of the representation of sentence i and the representation of the document up to sentence $i - 1$, $\mathbf{h}_i^\top \mathbf{W}_r \tanh(\mathbf{s}_i)$;
- **absolute position score:** a linear function of the embedding of the absolute position of the sentence, $\mathbf{W}_{ap} \mathbf{p}_i^a$;
- **relative position score:** a linear function of the embedding of the relative position of the sentence, $\mathbf{W}_{rp} \mathbf{p}_i^r$.

The probability to keep the i^{th} sentence in the summary is calculated as follows:

$$p(1|\mathbf{h}_i, \mathbf{s}_i, \mathbf{d}) = \sigma \left(\mathbf{W}_c \mathbf{h}_i + \mathbf{h}_i^\top \mathbf{W}_s \mathbf{d} + \mathbf{h}_i^\top \mathbf{W}_r \tanh(\mathbf{s}_i) + \mathbf{W}_{ap} \mathbf{p}_i^a + \mathbf{W}_{rp} \mathbf{p}_i^r \right), \quad (1)$$

where $\mathbf{d}$ is the representation of the whole document obtained by averaging all the hidden states $\{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n\}$, and $\mathbf{s}_i$ is the representation of the document up to the previous sentence, a weighted average of $\{\mathbf{h}_1, \dots, \mathbf{h}_{i-1}\}$:

$$\mathbf{s}_i = \sum_{j=1}^{i-1} \mathbf{h}_j p(1|\mathbf{h}_j, \mathbf{s}_j, \mathbf{d}). \quad (2)$$

3 User Interface

Table 1. Performance on the NYT corpus. The gain over SummaRuNNer is given after the + sign.

	ROUGE-1	ROUGE-L
SummaRuNNer	45.3	34.65
HeterSUMGraph	46.76 +2.0%	35.21 +1.6%
RadSum	46.91 +2.4%	35.35 +2.0%

For the purpose of the demonstration, we train our method, RadSum, on the New York Times annotated corpus [4]. Its performance in terms of ROUGE-1 and ROUGE-L scores is reported in Table 1, along with the scores achieved by SummaRuNNer and HeterSUMGraph. Fig. 2 shows the default interface of the proposed application. The left side of the windows allows to input the document to summarize and adjust general settings. The right side of the window shows the summary extracted from that document. For each sentence, the contribution of the 5 scores to the overall probability is depicted with a bar chart. Each sentence is colored according to the dominant score (scaled between 0 and 1 with the sigmoid function). When inputting a press article about the Nobel Prize in Physics awarded in October 2023, it produces a 4-sentence long summary that focuses on the laureates and the implications of their discovery by picking the 1st sentence, particularly for its position in the document, and sentences 4-6 mostly because of their content and salience. Fig. 3 shows how one could leverage the app to tailor the summary according to its needs. Another 4-sentence long summary focusing on the discovery itself is obtained by selecting sentences 19, 28, 29 and 31 solely based on the salience score (using the filter button). It also shows how one can manually filter the sentences: here, sentence 31 has been removed from the summary, which resulted in the automatic addition of sentence 30 (the next most salient sentence) to keep the summary at a length of 4 sentences.

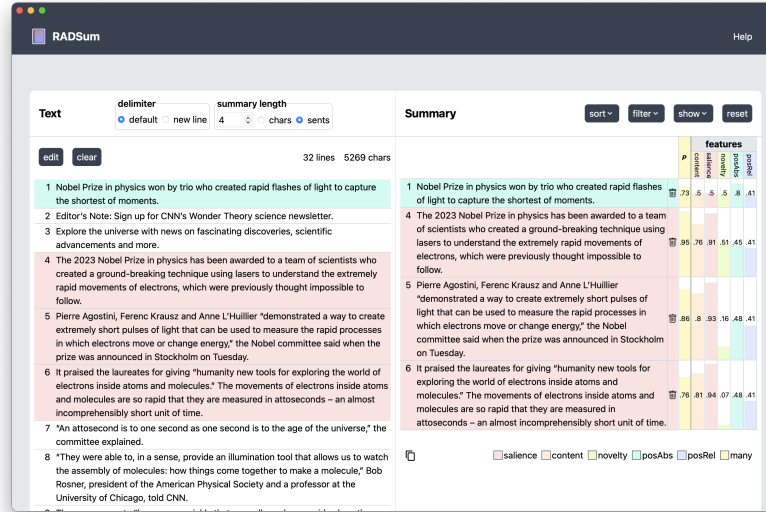


Fig. 2. Default interface. The P column (in yellow) corresponds to the probability to keep the sentence.

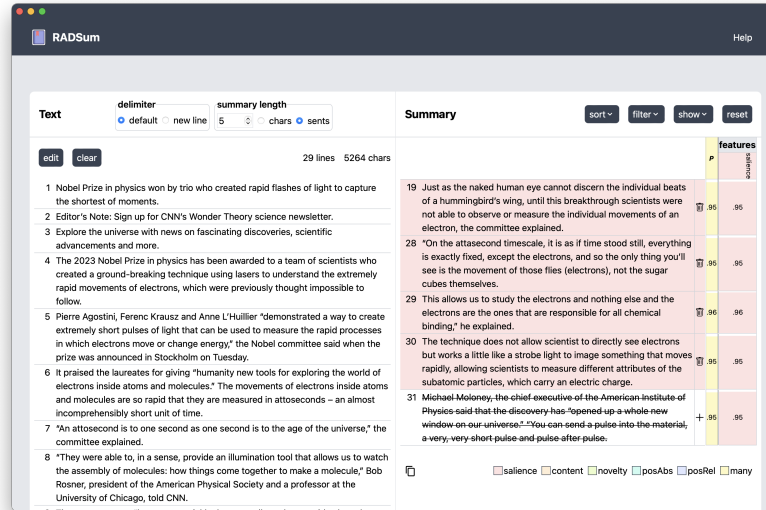


Fig. 3. Example of interaction: the user has selected salience as the only score and has removed sentence 31 from the summary.

References

1. Azaria, A., Azoulay, R., Reches, S.: Chatgpt is a remarkable tool – for experts (2023)
2. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? ICLR (2022)
3. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. EMNLP (2014)
4. Durrett, G., Berg-Kirkpatrick, T., Klein, D.: Learning-based single-document summarization with compression and anaphoricity constraints. ACL (2016)
5. Jentzsch, S., Turan, C.: Gender bias in BERT - measuring and analysing biases through sentiment rating in a realistic downstream classification task. GeBNLP workshop @ ACL (2022)
6. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. ACM Computing Surveys **55**(12) (2023)
7. Nallapati, R., Zhai, F., Zhou, B.: Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. AAAI (2017)
8. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Aspell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.: Training language models to follow instructions with human feedback. ArXiv Technical Report (2022)
9. Rozado, D.: The political biases of chatgpt. Social Sciences **12**(3) (2023)
10. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. ICLR (2018)
11. Wang, D., Liu, P., Zheng, Y., Qiu, X., Huang, X.: Heterogeneous graph neural networks for extractive document summarization. ACL (2021)