# Non-Parametric Memory Guidance for Multi-Document Summarization

Florian Baud, Alex Aussem

# Non-Parametric Memory Guidance for Multi-Document Summarization

**Florian Baud**
LIRIS, Villeurbanne
Visiativ
florian.baud@visiativ.com

**Alex Aussem**
LIRIS, Villeurbanne
alexandre.aussem@liris.cnrs.fr

## Abstract

Multi-document summarization (MDS) is a difficult task in Natural Language Processing, aiming to summarize information from several documents. However, the source documents are often insufficient to obtain a qualitative summary. We propose a retriever-guided model combined with non-parametric memory for summary generation. This model retrieves relevant candidates from a database and then generates the summary considering the candidates with a copy mechanism and the source documents. The retriever is implemented with Approximate Nearest Neighbor Search (ANN) to search large databases. Our method is evaluated on the *MultiXScience* dataset which includes scientific articles. Finally, we discuss our results and possible directions for future work.

## 1 Introduction

Multi-document summarization is performed using two methods: extractive (Wang et al., 2020; Liu et al., 2021) or abstractive (Jin et al., 2020; Xiao et al., 2022). So-called extractive methods rank sentences from source documents that best summarize them. These methods reuse important information well to construct a good summary but they lack coherence between sentences. To overcome this issue, abstractive methods are studied to imitate human writing behavior. They show great performance in human writing style but they often miss key information.

To make abstractive models aware of essential information, (Dou et al., 2021) guides their model with additional information like a set of keywords, graph triples, highlighted sentences of source documents, or retrieved similar summaries. Their method, which uses every guidance previously mentioned, improves summary quality and controllability compared with unguided models. However, guidances require specific training data, especially for keywords, graph triples, and highlighted sentences.

Our proposal is that by guiding with pre-existing summaries, the model can draw inspiration from the summary as a whole. But also be able to extract keywords and phrases using a copy mechanism. Consequently, this work focuses on guidance by similar summaries extracted from a knowledge base using a similarity metric between source documents and pre-existing summaries. The model, inspired by RAG (Lewis et al., 2020), is fully differentiable. In addition, the model generator uses a copy mechanism on the candidates returned from the knowledge base, inspired by (Cai et al., 2021). The findings of these two studies motivated the development of our model for the multi-document text summarization task.

We demonstrate the potential of our method on *MultiXScience* (Lu et al., 2020). This dataset gathers scientific articles where we have to generate the *"related work"* part with the *"abstract"* of the source article and the *"abstracts"* of the citations. In the case of scientific articles, we believe that the source documents are insufficient to generate the *"related work"* part because external knowledge is necessary to write such a paragraph.

In this work, we investigate a sequence-to-sequence model guided by a memory retriever of similar summaries. Specifically, source documents are the input of the memory retriever, which returns the top k similar summaries from a potentially large database using an approximate nearest neighbor search. Then, the decoder generates the summary taking into account the source and retrieved summaries and is trained to identify interesting texts for the targeted summary. The code of our work is available on GitHub[1].

[1] https://github.com/florianbaud/retrieval-augmented-mds (visited on 11/08/2023)



*Proceedings of Recent Advances in Natural Language Processing*, pages 153–158
Varna, Sep 4–6, 2023
https://doi.org/10.26615/978-954-452-092-2_017

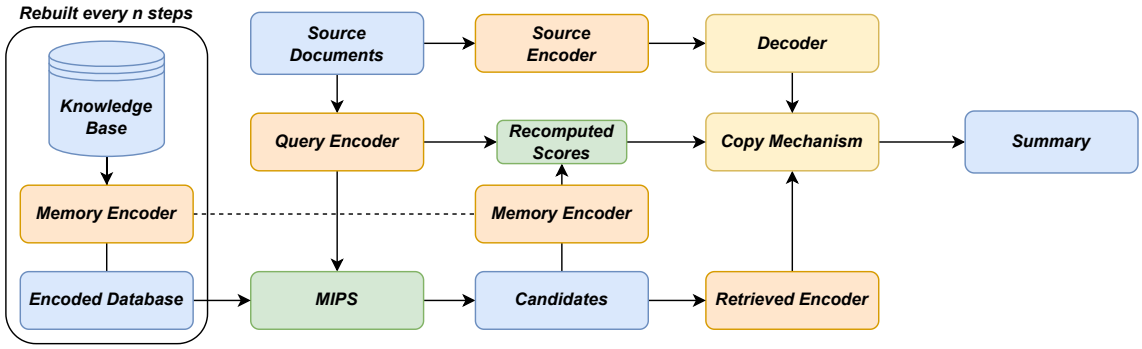


Figure 1: In the first step, the knowledge base is built by encoding all documents with the memory encoder. Then the source documents are transformed with a query encoder and with a source encoder, the query encoder is used to search the knowledge base. The encoded source is used to represent the source documents for the generation of the summary. After retrieving the top-$k$ of the search, they are encoded with the retrieved encoder and again with the memory encoder to recalculate the relevance score for back-propagation. Then, the decoder takes as input the source documents and the relevant documents for the generation of the summary.

Our contribution is twofold: firstly, we integrate a retriever to retrieve candidates for the generation of the summary, and secondly, we make use of a copy mechanism to incorporate these candidates into the generation procedure.

## 2 Related Work

We start with a brief review of related work. (Cohan et al., 2018) proposes to capture the structure of the document to better represent the information of the source document. Their method is applied to scientific articles from *Arvix* and *Pubmed* which are long documents. For the same purpose, (Cohan and Goharian, 2018; Yasunaga et al., 2019) propose to generate a summary from the articles that cite the article to be summarised. The disadvantage of these methods is that they cannot be used when writing an article. In this work, we use references and not the papers that cite the documents to be summarised. More recently, (Xiao et al., 2022) proposed a pre-training strategy dedicated to multi-document text summarisation, their masking strategy showed significant improvement for the MDS task. They applied their method to the *MultiXScience* dataset.

The models using guidances are close to our work, indeed (Cao et al., 2018; Dou et al., 2021) use retrieved summaries to better control the summary generation. However, they use information retrieval systems such as *ElasticSearch* to find candidates for summary generation. Also, (An et al., 2021) has introduced dense search systems for text summarization, but they do not train the retriever with the summary generator. In our case, the retriever is dense and trainable to find the most relevant candidates for the generation of the summary.

In addition, retrieval-augmented models share commonalities with our work. RAG, (Lewis et al., 2020) which introduced this type of model, is used for the question-answering task, where a context is given to answer the question. The model retrieves several contexts with a retriever and then answers the question using each of the retrieved candidates. These types of models are also used in the translation task, where (Cai et al., 2021) translates a sentence with a pre-established translation base. Their model searches this base for translations close to the sentence to be translated and then incorporates them into the generation of the translation through a copy mechanism. This approach shares some similar intuition with our proposed approach because our architecture is based on an augmented retriever that incorporates the memory by means of a copy mechanism. It is interesting to investigate whether the encouraging success of the copy mechanism recently obtained in translation carries over to the MDS task.

## 3 Proposed Method

Inspired by (Cai et al., 2021), we propose a model composed of a memory retriever and a copy generator. Figure 1 illustrates our framework, where we start by encoding the entire knowledge base. After an arbitrary number of steps during the training, the encoded knowledge base is updated. Then, the forward pass encodes source documents and finds similar documents. Retrieved documents are encoded and fed to the generator with the source documents.

Our memory retriever has multiple encoders, one for encoding the query, one for the knowledge base, one for the sources documents, and one for the retrieved candidates. Our copy generator is a decoder with a cross-attention mechanism on source document embeddings and a copy mechanism on retrieved candidates, which is placed at the top of the decoder. We begin describing the retriever and then show how our generator works.

### 3.1 Memory Retriever

The retrieval approach consists of source documents as query and documents from a knowledge base denoted respectively by $q$ and $c$. Documents are often too long to be encoded with a *Transformer* (Vaswani et al., 2017), so we used a *LongFormer* (Beltagy et al., 2020) model. *LongFormer* has a Transformer-like architecture that can deal with long input sequences by attending tokens with windowed attention and global attention on a few tokens. We encode source documents and candidates documents with a pretrained *LongFormer* model separated by a special token (*[DOC]*) :

$$h^q = LED^q_{enc}(q)$$
$$h^m = LED^m_{enc}(m)$$

where the *LongFormer* encoder is denoted by $LED_{enc}$. All documents in the knowledge base are encoded and stored in an index. For retrieving candidates, we take the *[CLS]* token of encoders output that we normalize and we define a relevance function :

$$h^q_{cls} = norm(h^q_{cls})$$
$$h^m_{cls} = norm(h^m_{cls})$$
$$score(x, y) = x^\top \cdot y$$

We then calculate the relevance score on normalized tokens, which represents the cosine similarity between source documents $q$ and candidate documents $m$ that fall in the interval $[-1, 1]$.

For fast retrieval, we retrieve the top-$k$ candidates $m_{topk} = (m_1, \ldots, m_k)$ using the maximum inner product search (MIPS) implemented with FAISS (Johnson et al., 2021). At each training step, we calculate the actual embedding of candidates $\{h^m_{cls,i}\}^k_{i=1}$ and compute their relevance scores $\{s_i = score(h^m_{cls,i}, h^q_{cls})\}^k_{i=1}$ for back-propagation as in (Cai et al., 2021; Lewis et al., 2020). The recalculated score biases the decoder copy mechanism , which we detail in section 3.2.

The memory encoder does not re-encode all the knowledge base at each training step because this would be expensive computation. Instead, the knowledge base and the MIPS index are updated at regular intervals defined arbitrarily. On the other hand, we encode the retrieved top-$k$ candidates and the source documents with two encoders, $LED^r_{enc}$ and $LED^s_{enc}$, as shown below:

$$h^s = LED^s_{enc}(q)$$
$$h^r_{topk} = LED^r_{enc}(m_{topk})$$

These two results are forwarded to the copy generator, which we detail in the next section.

### 3.2 Copy Generator

In the generation part of our model, we use the decoder from *LongFormer* and apply a copy mechanism to previously retrieved candidates. Formally, we have :

$$h^d = LED_{dec}(y, h^s)$$

where $LED_{dec}$ corresponds to the decoder part of the *LongFormer* model, and $y$ is the targeted summary. The decoder attends over source documents $h^s$ and previous tokens $y_{1:t-1}$, producing a hidden state $h^d_t$ at each time step $t$. The probability of the next token is calculated with a $softmax$ function:

$$P_{dec}(y_t) = softmax(W_d \cdot h^d_t + b_d) \quad (1)$$

where $W_d$ is a $hiddens_{size} \times vocab_{size}$ matrix and $b_d$ is the bias; both are trainable parameters.

Then, we incorporate the top-$k$ candidates $m_{topk}$ with a copy mechanism by calculating a cross attention between $h^d_t$ and $h^r_{topk}$. To this end, we reuse the cross-attention part of *LongFormer* to add it after its original decoder. This new layer has only one attention head in order to use the attention weights as the probability to copy a word from top-$k$ candidates.

Given $k$ documents encoded in $h^r_{topk}$, then we can construct a set of token embedding $\{r_{i,j}\}^{L_i}_{j=1}$ where $i \in [1, k]$, $j \in [1, L_i]$ and $L_i$ is the length of document $i$. Formally, the attention weight of the $j$th token in the $i$th relevant document is expressed as,

$$\alpha_{ij} = \frac{\exp(h^{d\top}_t W_a r_{i,j} + \beta s_i)}{\sum^k_{i=1}\sum^{L_i}_{j=1}\exp(h^{d\top}_t W_a r_{i,j} + \beta s_i)}$$
$$c_t = W_c \sum^k_{i=1}\sum^{L_i}_{j=1} \alpha_{ij} r_{i,j}$$

where $\alpha_{ij}$ is the attention weight of the $j$th token in the $i$th relevant document, $W_a$ and $W_c$ are learnable parameters, $c_t$ is a weighted representation of top-$k$ candidates and $\beta$ is a learnable scalar that controls the relevance score between the retrieved candidates and the decoder hidden state, enabling the gradient flow to the candidates encoders as in (Cai et al., 2021; Lewis et al., 2020). Equation 1 may be rewritten to include the memory:

$$P_{dec}(y_t) = softmax(W_d \cdot (h_t^d + c_t) + b_d) \quad (2)$$

Thus the next token probability takes into account the attention weights of the top-$k$ candidates. The final next token probability is given by:

$$P(y_t) = (1 - \lambda_t)P_{dec}(y_t) + \lambda_t \sum_{i=1}^{k} \sum_{j=1}^{L_i} \alpha_{ij} \mathbb{1}_{r_{ij}=y_t}$$

where $\lambda_t$ is a gating scalar computed by a feed-forward network $\lambda_t = g(h^d, c_t)$. The model is trained with the log-likelihood loss $\mathcal{L} = -\log P(y^*)$ where $y^*$ is the target summary.

### 3.3 Training Details

Our model is composed of several encoders and one decoder based on the *LongFormer* (Beltagy et al., 2020) large model. Therefore, the size of our model attains 1.9B of trainable parameters. Then we used the *DeepSpeed* (Rasley et al., 2020) library for the training. Our model uses the *LongFormer* pretrained models available on *HuggingFace*[2].

The training of the model makes use of *MultiXScience* data comprising 30,369 scientific articles for training, 5,066 validation, and 5,093 test articles. The objective is to generate the related work using the abstract of the article and the abstracts of the cited articles. This is an interesting dataset to experiment with because writing a related work part requires knowledge beyond the scope of the source documents.

**Cold start problem** At the beginning of the training, the weights are randomly initialized. Therefore the retriever selects low-quality candidates that don't send out a good signal for training. Under these conditions, the retriever cannot improve, and the model will ignore the retriever's candidates. To overcome this cold start problem, we pre-trained the retriever on the *MultiXScience* data to improve the quality of the retriever. The objective is to maximize the similarity between the abstract and the related work section. These two sections are encoded with the two encoders of the retriever to calculate the cosine similarity.

In concrete terms, pre-training works as follows. For a batch size equal to $N$, we have $N$ "abstract" sections encoded with $A = \{LED^q_{enc}(a_i)\}_{i=1}^N$ and $N$ "related work" sections encoded with $B = \{LED^m_{enc}(b_j)\}_{j=1}^N$, in order to obtain a cosine similarity equal to 1 when $j = i$ corresponds to positive examples and -1 otherwise for negative examples. We calculate for each element in $A$, the following errors:

$$\mathcal{L}_i(A, B) = -\log \frac{\exp\left(score(A_i, B_i)/\tau\right)}{\sum_{j=1}^{N} \exp\left(score(A_i, B_j)/\tau\right)}$$

where $\tau$ is an arbitrarily chosen temperature parameter. The final error is $\mathcal{L} = \sum_{i=1}^{N} \mathcal{L}_i$ back-propagated in the two encoders of the retriever.

## 4 Experiments

In this section, we report on the experiments performed on the *MultiXScience* dataset to evaluate our model. Training the full model is more difficult due to its size but also due to the cold start problem. The latter corresponds to the fact that the similar summaries retrieved are not sufficiently relevant to help the model. In addition, we have trained two other methods adapted to text summarisation as a comparison, *Bart* (Lewis et al., 2019) and *T5* (Raffel et al., 2020). We detail the training procedure for each of them. All models use the beam search method to generate summaries. We chose a beam size of 4, a length penalty of 1.0, and limited the repetition of tri-grams. The rouge scores (Lin, 2004) on the *MultiXScience* dataset are reported in table 1.

| Method | R-1 | R-2 | R-L |
|---|---|---|---|
| Ours | 30.6 | 6.5 | 17.7 |
| Bart (Our run) | 32.4 | 7.2 | 17.3 |
| T5 (Our run) | 29.6 | 6.3 | 17.0 |
| Primera* | 31.9 | **7.4** | 18.0 |
| PointerGenerator* | **33.9** | 6.8 | **18.2** |

Table 1: The ROUGE score (R-1/R-2/R-L) of our preliminary results on the *MultiXScience* test dataset. The * symbol means that the results have been borrowed from (Xiao et al., 2022).

[2]https://huggingface.co/allenai (visited on 11/08/2023)

**Reduced model** To reduce the computational burden, we used a reduced model where the knowledge base is not reconstructed. In addition, the memory encoder parameters were frozen in order to reduce the complexity of the training. These two modifications reduced the training time considerably. Indeed, the burden of reconstructing the knowledge base was overwhelming. The reduced model has fewer trainable parameters (1.4B). The model was trained for 12.000 steps on four v100 GPUs with Adam optimizer and a learning rate of $3e-5$, a batch size of 64, a top-$k$ of 5 for the retriever, and with 2.000 warmup steps and linear decay. Despite its reduction in size, we observe that the model is competitive with the state of the art.

**Bart** We fine-tuned a *Bart-large* model on the *MultiXScience* dataset using a single v100 GPU over two days. The model weights were updated for 20,000 steps with a learning rate of 3.0e-5. A linear warmup for 2,000 steps was applied to the learning rate. We also limited the norm of the gradient to 0.1. The training aims to minimize cross-entropy with a smoothing label of 0.1. The *MultiXScience* articles have been concatenated using the '\n\n' separator. The results show that Bart is competitive with the state of the art.

**T5** The *T5-large* model was fine-tuned on the same dataset as before. The training lasted 4 days on a single v100 GPU, this model is slightly larger and was trained in fp32 precision. As T5 is a text-to-text model, we have used the prefix 'summarize:' for the input documents, which are separated by the separator '\n\n'. The model was trained for 7,000 steps with a learning rate of 1.0e-4 and a batch size of 64. A linear warm-up of up to 2000 steps and a gradient norm limitation of 0.1 was applied. The error to be minimized is the cross-entropy with a label smoothing of 0.1.

## 5 Conclusion and Future Work

This paper presents an architecture for multi-document text summarization inspired by retrieval-augmented models. This architecture includes a retriever that searches a knowledge base to find relevant documents for the generation of a summary. These documents are integrated in the generation by means of a copy mechanism. A reduced version of the model was evaluated on the *MultiXScience* dataset. The preliminary results are already competitive with the state of the art however we expect to improve our results further by: 1) properly fixing the cold start problem, and 2) training the full model. In the future, we also plan to increase the size of the knowledge base with new data and apply our method to other MDS benchmark datasets.

## Acknowledgments

We gratefully acknowledge support from the CNRS/IN2P3 Computing Center (Lyon - France) for providing computing and data-processing resources needed for this work. In addition, This work was granted access to the HPC resources of IDRIS under the allocation 2022-AD011013300 made by GENCI. Finally, we would like to thank Roch Auburtin from Visiativ for his advice.

## References

Chenxin An, Ming Zhong, Zhichao Geng, Jianqiang Yang, and Xipeng Qiu. 2021. Retrievalsum: A retrieval enhanced framework for abstractive summarization.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. 2021. Neural machine translation with monolingual translation memory. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7307–7318, Online. Association for Computational Linguistics.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Arman Cohan and Nazli Goharian. 2018. Scientific document summarization via citation contextualization and scientific discourse. *International Journal on Digital Libraries*, 19(2):287–303.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. GSum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online. Association for Computational Linguistics.

Hanqi Jin, Tianming Wang, and Xiaojun Wan. 2020. Multi-granularity interaction network for extractive and abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6244–6254, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

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.

Ye Liu, Jianguo Zhang, Yao Wan, Congying Xia, Lifang He, and Philip Yu. 2021. HETFORMER: Heterogeneous transformer with sparse attention for long-text extractive summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 146–154, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yao Lu, Yue Dong, and Laurent Charlin. 2020. Multi-XScience: A large-scale dataset for extreme multi-document summarization of scientific articles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8068–8074, Online. Association for Computational Linguistics.

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.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

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.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online. Association for Computational Linguistics.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2022. PRIMERA: Pyramid-based masked sentence pre-training for multi-document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5245–5263, Dublin, Ireland. Association for Computational Linguistics.

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander R. Fabbri, Irene Li, Dan Friedman, and Dragomir R. Radev. 2019. Scisummnet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7386–7393.