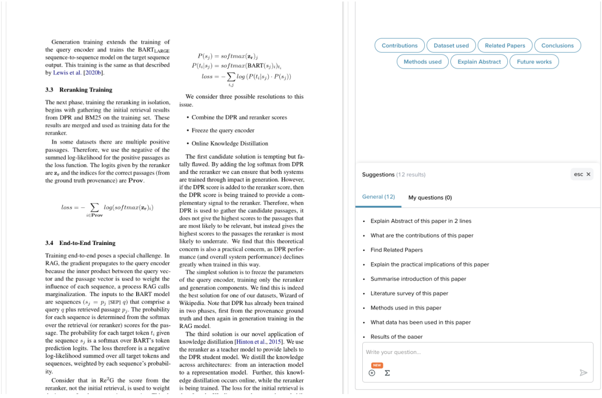
SciSpace Copilot: Empowering Researchers through Intelligent Reading Assistance

Trinita Roy, Asheesh Kumar, Daksh Raghuvanshi, Siddhant Jain, Goutham Vignesh, Kartik Shinde, Rohan Tondulkar

SciSpace
(trinita,asheesh,daksh,siddhant,goutham,kartik,rohan)@typeset.io

Abstract

We introduce SciSpace Copilot, an AI research assistant that helps in understanding and reading research papers faster by providing a plethora of features. Answering questions from a document has recently become popular using the Retrieval Augmented Generation (RAG) approach. Our tool uses an advanced question-answering pipeline to give accurate answers while also citing sources from the paper. We also provide other valuable features on scientific text, including generating explanations, generating summaries, adding notes and highlights, and finding related papers from our 280 million corpus. Our tool supports 75+ languages, making research more accessible across language barriers. Thousands of users use SciSpace Copilot on a daily basis by uploading their articles to understand research faster and better. Our tool can be accessed at this link: https://typeset.io.



Generation training extends the training of the query encoder and trains the $\text{BART}_{\text{LARGE}}$ sequence-to-sequence model on the target sequence output. This training is the same as that described by Lewis et al. [2020b].

3.3 Reranking Training

The next phase, training the reranking in isolation, begins with gathering the initial retrieval results from DPR and BM25 on the training set. These results are merged and used as training data for the reranker.

In some datasets there are multiple positive passages. Therefore, we use the negative of the summed log-likelihood for the positive passages as the loss function. The logits given by the reranker are $\mathbf{z_r}$ and the indices for the correct passages (from the ground truth provenance) are $\mathbf{Prov}$.

$$loss = -\sum_{i \in \mathbf{Prov}} log(softmax(\mathbf{z_r})_i)$$

3.4 End-to-End Training

Training end-to-end poses a special challenge. In RAG, the gradient propagates to the query encoder because the inner product between the query vector and the passage vector is used to weight the influence of each sequence, a process RAG calls marginalization. The inputs to the BART model are sequences ($s_j = p_j$ [SEP] q) that comprise a query q plus retrieved passage p_j. The probability for each sequence is determined from the softmax over the retrieval (or reranker) scores for the passage. The probability for each target token t_i given the sequence s_j is a softmax over BART's token prediction logits. The loss therefore is a negative log-likelihood summed over all target tokens and sequences, weighted by each sequence's probability.

Consider that in Re^2G the score from the reranker, not the initial retrieval, is used to weight

$$P(s_j) = softmax(\mathbf{z_r})_j$$
$$P(t_i|s_j) = softmax(\text{BART}(s_j)_i)_{t_i}$$
$$loss = -\sum_{i,j} log\left(P(t_i|s_j) \cdot P(s_j)\right)$$

We consider three possible resolutions to this issue.

- Combine the DPR and reranker scores
- Freeze the query encoder
- Online Knowledge Distillation

The first candidate solution is tempting but fatally flawed. By adding the log softmax from DPR and the reranker we can ensure that both systems are trained through impact in generation. However, if the DPR score is added to the reranker score, then the DPR score is being trained to provide a complementary signal to the reranker. Therefore, when DPR is used to gather the candidate passages, it does not give the highest scores to the passages that are most likely to be relevant, but instead gives the highest scores to the passages the reranker is most likely to underrate. We find that this theoretical concern is also a practical concern, as DPR performance (and overall system performance) declines greatly when trained in this way.

The simplest solution is to freeze the parameters of the query encoder, training only the reranker and generation components. We find this is indeed the best solution for one of our datasets, Wizard of Wikipedia. Note that DPR has already been trained in two phases, first from the provenance ground truth and then again in generation training in the RAG model.

The third solution is our novel application of knowledge distillation [Hinton et al., 2015]. We use the reranker as a teacher model to provide labels to the DPR student model. We distill the knowledge across architectures: from an interaction model to a representation model. Further, this knowledge distillation occurs online, while the reranker is being trained. The loss for the initial retrieval is

- Explain Abstract of this paper in 2 lines
- What are the contributions of this paper
- Find Related Papers
- Explain the practical implications of this paper
- Summarise introduction of this paper
- Literature survey of this paper
- Methods used in this paper
- What data has been used in this paper
- Results of the paper

Figure 1: Copilot feature window preview

Introduction

Researchers today are facing a big problem of keeping up-to-date with latest research as approx. 100 papers are published daily in machine learning domain itself. They want to extract important insights from a publication and get answers to some important questions from the publication. Another problem is the trouble of understanding complex parts of research papers which requires referring to multiple resources. This increases the overall time of reading a paper, leading to less research data being consumed. Available existing tools don't address these problems.

Recent advancements in large language models (LLMs) have achieved remarkable progress in various tasks, displaying enhanced performance and versatility in answering questions across diverse domains. LLMs have been trained on terabytes of open access knowledge but are known to generate misinformation with inaccurate citations making it difficult to trust outputs from their own knowledge. The approach outlined in the paper (Lewis et al. 2020; Cai et al. 2022) introduces a novel approach that employs retrieval-augmented generation (RAG) models. These models combine pre-trained parametric and non-parametric memory elements with the objective of improving language generation. As a result, they achieve state-of-the-art results on open domain QA tasks responses that are more precise, varied, and fact-based. (Nguyen, Le, and Nguyen 2022) developed a "retriever-reader" framework for querying scientific texts, employing a sliding window technique to manage large context scales in documents by dividing them into manageable text blocks.

Our paper demonstrates an AI-powered Copilot assistant which has the following characteristics:

- Provide evidence-based responses and clarify intricate research elements, including texts and equations, facilitating comprehensive understanding.
- Streamline research efforts through content summarization, support for regional languages, and improved engagement with highlighting, note-taking, and personalized recommendations.

Methodology

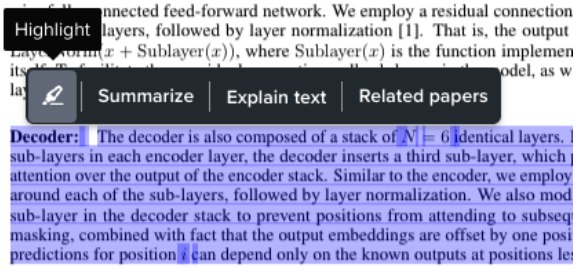
This section gives details about the core components forming the building blocks of features in SciSpace Copilot:

- **Parsing**: We use various parsing tools like Grobid[1], PyMuPDF[2] and Science-Parse[3] to get a clean textual representation for research pdfs.

[1] https://github.com/kermitt2/grobid

[2] https://github.com/pymupdf/PyMuPDF

[3] https://github.com/allenai/science-parse



nnected feed-forward network. We employ a residual connection
layers, followed by layer normalization [1]. That is, the output
$(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemen
its odel, as w
la

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. I
sub-layers in each encoder layer, the decoder inserts a third sub-layer, which
attention over the output of the encoder stack. Similar to the encoder, we employ
around each of the sub-layers, followed by layer normalization. We also mod
sub-layer in the decoder stack to prevent positions from attending to subseq
masking, combined with fact that the output embeddings are offset by one posi
predictions for position i can depend only on the known outputs at positions le

Figure 2: Functionalities on text selection.

- **Embedding generation**: We use state-of-the-art open-source embedding models to generate embeddings on chunks of text data from the parsing output. These embeddings are stored in a vector database for later use in various features.
- **Context extraction**: We use structured and unstructured approaches to extract relevant context for a given question. We parse through the parsing outputs to give importance to the structure of the paper and also use vector search to get relevant text.
- **LLMs and prompt engineering**: We use generative LLMs to generate required output for each use case. We use tools such as OpenAI (OpenAI 2023), Anthropic [4], Bard [5], etc., as well as open source models for certain use cases. Prompt engineering (White et al. 2023; Oppenlaender, Linder, and Silvennoinen 2023; Sorensen et al. 2022) is done to get the desired outputs. Relevant contexts are passed from the research paper to get factual responses and reduce the possibilities of hallucination (Li et al. 2023; McKenna et al. 2023; Mündler et al. 2023).

Core Functionalities

Features on Text Selection

SciSpace Copilot gives the options of Explain text, Summarize, Related papers and highlight to users on selecting a piece of text in the paper as shown in Figure 2.

- **Summarize**: The ”Summarize” option will generate a short summary for a long piece of text to help users to read papers faster by getting summaries of each section.
- **Explaining complex text and equations**: Users can use the ”Explain text” option to understand research paper text and use the snip tool to understand formulae, equations, and tables by selecting the area containing the required information as shown in Figure 3.
- **Related papers**: This option recommends papers from a corpus of 200+ million research papers similar to the selected text which helps in discovery of more papers.
- **Highlight**: Users can apply colorful highlights and add accompanying notes to any uploaded research paper.

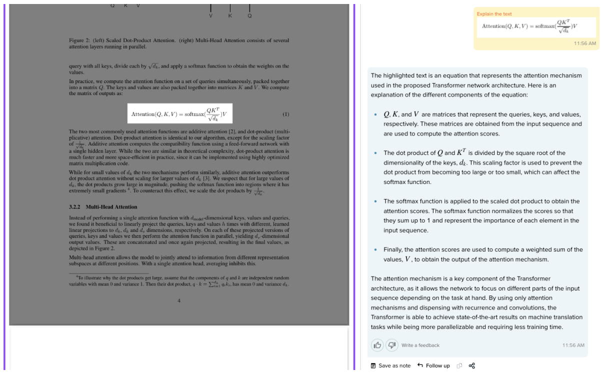


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

3.2.2 Multi-Head Attention

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

[4] To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k.

4

Explain the text

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

11:56 AM

The highlighted text is an equation that represents the attention mechanism used in the proposed Transformer network architecture. Here is an explanation of the different components of the equation:

- Q, K, and V are matrices that represent the queries, keys, and values, respectively. These matrices are obtained from the input sequence and are used to compute the attention scores.
- The dot product of Q and K^T is divided by the square root of the dimensionality of the keys, d_k. This scaling factor is used to prevent the dot product from becoming too large or too small, which can affect the softmax function.
- The softmax function is applied to the scaled dot product to obtain the attention scores. The softmax function normalizes the scores so that they sum up to 1 and represent the importance of each element in the input sequence.
- Finally, the attention scores are used to compute a weighted sum of the values, V, to obtain the output of the attention mechanism.

The attention mechanism is a key component of the Transformer architecture, as it allows the network to focus on different parts of the input sequence depending on the task at hand. By using only attention mechanisms and dispensing with recurrence and convolutions, the Transformer is able to achieve state-of-the-art results on machine translation tasks while being more parallelizable and requiring less training time.

Figure 3: Explanations based on selected texts, tables and equations.

Question Answering

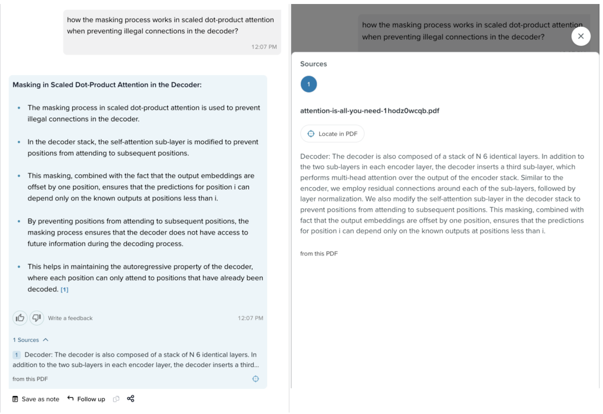
Users inquire about research paper concepts, receiving answers with precise citations via SciSpace Copilot. Users can locate the source in paper as shown in Figure 4. Follow-up questions are supported, and answers can be stored as Notes for future reference.

Regional Language Support

Copilot enables language-agnostic research paper comprehension. Users can ask and receive answers, as well as read papers, in their regional language among 75+ supported options.

Conclusion

SciSpace Copilot is a generative AI tool that streamlines the process of consuming research by rapidly distilling insights from papers and making complex content accessible to newer researchers. It saves time and increases research consumption with features like regional language support, which is particularly beneficial for users in Asia and Africa. The tool keeps researchers up-to-date and speeds up their work.



how the masking process works in scaled dot-product attention when preventing illegal connections in the decoder?

12:07 PM

Masking in Scaled Dot-Product Attention in the Decoder:

- The masking process in scaled dot-product attention is used to prevent illegal connections in the decoder.
- In the decoder stack, the self-attention sub-layer is modified to prevent positions from attending to subsequent positions.
- This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.
- By preventing positions from attending to subsequent positions, the masking process ensures that the decoder does not have access to future information during the decoding process.
- This helps in maintaining the autoregressive property of the decoder, where each position can only attend to positions that have already been decoded. [1]

1 Decoder: The decoder is also composed of a stack of N 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third...

from this PDF

how the masking process works in scaled dot-product attention when preventing illegal connections in the decoder?

Sources

1

attention-is-all-you-need-1hodz0wcqb.pdf

Decoder: The decoder is also composed of a stack of N 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

from this PDF

Figure 4: Questions answering with navigable citations.

[4] https://www-files.anthropic.com/production/images/Model-Card-Claude-2.pdf

[5] https://blog.google/intl/en-africa/products/explore-get-answers/an-important-next-step-on-our-ai-journey/

References

Cai, D.; Wang, Y.; Liu, L.; and Shi, S. 2022. Recent advances in retrieval-augmented text generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 3417–3419.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Li, J.; Cheng, X.; Zhao, W. X.; Nie, J.-Y.; and Wen, J.-R. 2023. HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. *arXiv e-prints*, arXiv–2305.

McKenna, N.; Li, T.; Cheng, L.; Hosseini, M. J.; Johnson, M.; and Steedman, M. 2023. Sources of Hallucination by Large Language Models on Inference Tasks. *arXiv preprint arXiv:2305.14552*.

Mündler, N.; He, J.; Jenko, S.; and Vechev, M. 2023. Self-contradictory Hallucinations of Large Language Models: Evaluation, Detection and Mitigation. *arXiv preprint arXiv:2305.15852*.

Nguyen, D.-H.; Le, N.-K.; and Nguyen, M. L. 2022. Exploring Retriever-Reader Approaches in Question-Answering on Scientific Documents. In *Asian Conference on Intelligent Information and Database Systems*, 383–395. Springer.

OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

Oppenlaender, J.; Linder, R.; and Silvennoinen, J. 2023. Prompting ai art: An investigation into the creative skill of prompt engineering. *arXiv preprint arXiv:2303.13534*.

Sorensen, T.; Robinson, J.; Rytting, C. M.; Shaw, A. G.; Rogers, K. J.; Delorey, A. P.; Khalil, M.; Fulda, N.; and Wingate, D. 2022. An information-theoretic approach to prompt engineering without ground truth labels. *arXiv preprint arXiv:2203.11364*.

White, J.; Fu, Q.; Hays, S.; Sandborn, M.; Olea, C.; Gilbert, H.; Elnashar, A.; Spencer-Smith, J.; and Schmidt, D. C. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.