Domain Adapter for Sentence Transformer Models: Semantic Search

Jhon Rayo Raúl de La Rosa Ana
{j.rayomm,c.delarosap,as.medinam1}@uniandes.edu.co

Abstract

In this paper, we propose a domain-specific adapter for sentence transformer models to enhance semantic search capabilities within network of scientific publications, authors and venues. Traditional search engines rely on lexical matches to retrieve information, often missing semantically relevant results. By utilizing embeddings, our approach preserves the semantic meaning of input data, allowing for more accurate and relevant search results. We fine-tune a sentence transformer model using a curated dataset of academic publications, resulting in significant performance improvements over the base model. The training process, leveraging PyTorch and Sentence Transformers, involved the creation of multiple relationships among authors, papers, venues, and fields of study. We deployed the model using NVIDIA Inference Server, achieving optimized performance with low latency. Furthermore, we implemented a Retrieval-Augmented Generation (RAG) agent using large language models to answer queries about the dataset, accessible via a Telegram bot named SciSemanticBot. Our results demonstrate the effectiveness of our approach in enhancing information retrieval tasks while maintaining the model's original capabilities.

1. Introduction

Information retrieval systems allow efficiently querying a large corpus to retrieve relevant results. Traditional search engines depend on lexical matches to build an inverted index that is then used when a query is to be processed [1]. However, an obvious disadvantage is that the system will incorrectly rank results that may be semantically relevant but, for instance, use different wording.

Semantic search aims to improve the relevance of the search results through a novel approach known as *vector search*. Instead of indexing textual content, the system uses numeric representations of the data that enables matching based on vector similarity. These representations, commonly referred to as *embeddings*, preserve the semantic meaning of the input data in a higher dimensional space so that related textual contents are clustered together [2].

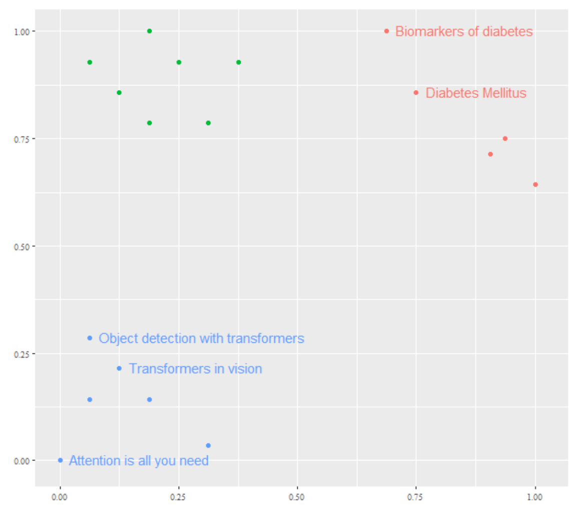


Figure 1: Sample text embeddings on a 2-dimensional space. Notice how sentences related to transformers are clustered together and are far apart from other clusters that encode different information such as diseases.

Text can be represented in various forms. A naive approach consists of creating a vector where each dimension corresponds to the frequency of a *n-gram* in the corpus [3]. However, such a model does not hold any semantic meaning, even if it may capture the position of words in the language. In contrast, semantic representations use machine learning models to extract the text *embeddings*. In particular, it has been found that pre-trained language models can be fine-tuned to provide high-quality representations of the textual data for either words, sub-words, n-grams, sentences, or whole paragraphs [4].

Once the *embeddings* have been computed, the system will determine the most relevant entries in the corpus using a measure of similarity such as cosine similarity. An exact computation is generally unfeasible since it would require a quadratic number of operations in terms of the number of entries in the corpus. Instead, retrieval systems often implement approximation algorithms to return the k nearest neighbors of the query *embedding*, Hierarchical Navigable Small World Graphs being a novel one [5].

An alternative approach to semantic search is based on knowledge bases which capture real-world relationships in the form of triples (e.g., subject-predicate-object) [6]. These systems are suitable for open-domain question-answering applications and hence are broadly used in different domains including healthcare, finance, academia, among others [3]. *SPARQL* is the standard language for querying RDF data [7] that is able to retrieve and manipulate data stored in the semantic database. However, this would require further enhancements to be able to process inputs in the form of natural language queries. Therefore, we propose to design a system that can leverage the knowledge base data to return relevant results via a *vector search* using fine-tuned textual embedding models for a particular domain.

One of the biggest challenges in generating IR systems using vector representations is that pre-trained models are often trained on general-purpose datasets. However, this knowledge is frequently insufficient to accurately represent the specific information relevant to each problem. Furthermore, given the current trend where the best-performing models are not open, there arises a need to implement solutions that optimally adjust the embedding model's domain to the particular case [2].

In this paper, we fine-tune a sentence transformer model through a transfer learning approach using triples from a knowledge base consisting of curated data from academic publications such as articles, authors, and venues, to provide relevant search results to user queries. Additionally, we present a comparative analysis of the accuracy of the model, showing significant improvement over the base model.

2. Background

Sentence transformer models are specifically trained to obtain sentence *embeddings* for natural language processing tasks such as textual similarity. They have shown robust performance for general semantic tasks. However, domain specific sentence transformer models most often outperform the generic model in domain specific scenarios [4].

all-MiniLM-L12-v2 is a generalized sentence transformer model trained on over 1 billion pairs of highly correlated sentences. The authors fine tuned a pre-trained model MiniLM using a self-supervised contrastive learning objective. While this model performs quite well in general scenarios, its performance may be improved for domains of interest.

Commonly, all of the weights of the sentence transformer model are updated during fine tuning to obtain the domain sentence *embeddings*. While this approach has shown to be optimal, it requires an extensive use of computational resources. Therefore, Tim Schopf et al. introduced the concept of domain adapters for sentence transformer models where they add a new layer of computation to the base model and the rest of the parameters remain fixed. As a result, the pre-training is not only much faster, but they also achieve similar results as the full pre-trained models. [4].

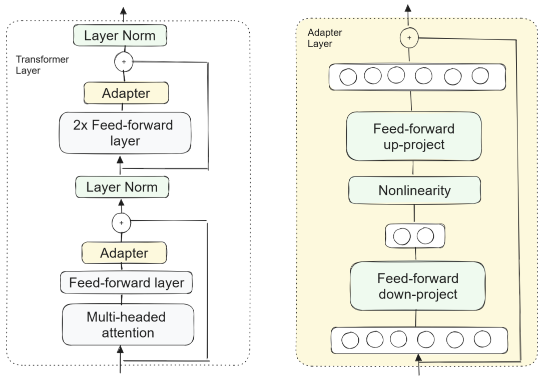


Figure 2: Houlsby-Adapter for transformer architectures.

Houlsby-Adapters use a bottleneck architecture that can be applied in transformer architectures. Each layer of a transformer contains two sub-layers: an attention layer and a feed-forward layer. The adapter layer is introduced in both of these sub-layers before the residual connection is added as illustrated in figure 2. The adapter consists of a first feed-forward layer that reduces the dimensionality of the input data and then uses another feed-forward layer to project it back to the original dimensions. In practice, this retains the most relevant pieces of information, while keeping the number of total trainable parameters low [8].

While the adapter architecture retains the performance of the base model and requires only a few new trainable parameters, it has a major disadvantage as they require access to the architecture and weights of the base model. In recent years, Large Language Models have gained vast popularity. Unfortunately, the largest models known to date are proprietary and are only made accessible via inference APIs, which creates numerous constraints.

Jinsung Yoon et al. introduced an alternative adapter *Search Adapter* that does not require access to the parameters of the base model and results in performant *embeddings* specifically designed for retrieval information tasks [2].

3. Domain Adapter Architecture

We present a domain adapter architecture that does not require modifications to the internals of the base sentence transformer model nor does it require fine-tuning of its parameters.

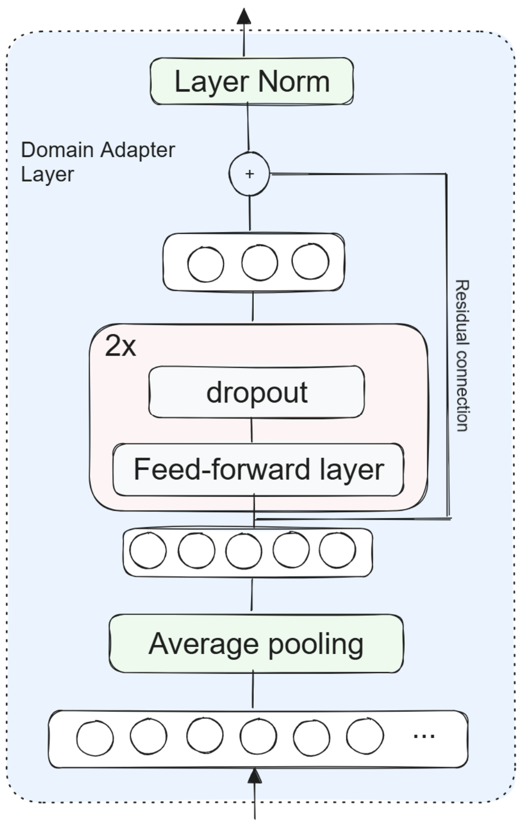


Figure 3: Domain adapter architecture.

The model, shown in figure 3, receives as input the generic sentence *embedding* from the base language model and then adds an average pooling layer that ensures the output *embedding* is of fixed length. At the core, it consists of n layers, each made up of a feed-forward layer followed by a dropout layer. This portion of the architecture acts as an encoder for the text *embedding*, capturing the most meaningful aspects of the data according to the domain. A residual connection is also included, which is added to the output of the previous layers. Finally, a normalization layer ensures that the resulting *embeddings* are of unit length.

For our use case, the *embeddings* obtained after the pooling layer consist of vectors with 384 dimensions. We then add 3 core layers where the first one projects the data to a higher dimensional space with 1024 features. The second layer reduces the dimensionality to 512 features, and the final layer brings it back to 384 features, matching the number of dimensions of the input data. The parameters of the base language model remain fixed throughout the entire training phase, as fine-tuning these large language models can be computationally intensive. Consequently, the total number of trainable parameters in the adapter module is only $1,116,032$, including biases and the residual coefficient.

The entire architecture implementation was carried out using PyTorch and the SentenceTransformer class provided by the library of the same name. The primary reason for this choice was to fully leverage the extensive framework proposed and already implemented by Sentence Transformers, which is currently one of the most popular in the industry. This library is aligned with Hugging Face, the largest open repository of language models at present. We found it particularly advantageous to utilize the loss and evaluation functions, as well as the training, inference, and instantiation methods for embedding models that Sentence Transformers offer. This integration allowed us to streamline our development process and ensure compatibility with state-of-the-art language models.

4. Training

The training was conducted on an NVIDIA A40 GPU with 24GB of memory using the SentenceTransformer methods for model fine-tuning. We employed the MultipleNegativesSymmetricRankingLoss function, which aims to maximize the similarity between positive pairs while minimizing it between negative pairs, effectively optimizing the model's ability to differentiate between relevant and irrelevant results.

Additionally, the InformationRetrievalEvaluator was used during training to measure the model's performance in retrieving relevant documents based on user queries. This evaluator computes metrics such as Mean Reciprocal Rank (MRR) and Recall@k, providing a comprehensive assessment of the model's retrieval capabilities.

The training process spanned 500 epochs with a batch size of 256. To further enhance training efficiency, we incorporated parameters such as warmup steps and Automatic Mixed Precision (AMP), a method used in deep learning to improve the computational efficiency through the use of single-precision and half-precision floating-point arithmetic [9]. Warmup steps help in gradually increasing the learning rate during the initial phase of training, thereby stabilizing the optimization process. The use of AMP leverages mixed-precision training to reduce memory usage and increase computational speed, facilitating faster and more efficient training on the GPU.

The combination of these methods and parameters ensured an effective and optimized fine-tuning process, resulting in a domain-specific sentence transformer model with improved accuracy and retrieval performance.

4.1. Dataset

To tailor the model to the domain of the scientific network encompassing publications, authors, and venues, we opt to utilize the REST API offered by Semantic Scholar. This solution allows for easy searches with a limit of up to one request per second if an API key is available. Requests and response storage were handled using Python scripts. The information provided by Semantic Scholar includes references and citations for each publication, as well as details of the authors, journals, areas of study, PDF URLs, and more.

A sample of the resulting data was extracted to define a series of triples that link the retrieved entities (authors, venues, papers) through their respective properties, such as cited/referenced in, published in/by, written by, co-authored with, etc. Once these triples were established, the training set was generated in a question-and-answer format, which is optimal for fine-tuning the embedding model to a specific domain. Below is a detailed description of the methodology followed to generate the datasets:

- **Author-Paper Relationships**: To create (author, wrote, paper) triples, the authorship information was extracted from the dataset. Each author's name and aliases were processed and cleaned for consistency. These details were used to form triples, including the author's name, paper title, author ID, and paper ID. For instance:
  - Triple: (N. Flyer, wrote, Solving PDEs with radial basis functions)
  - QA Pair: *"Who wrote the paper titled 'Solving PDEs with radial basis functions'? $\rightarrow$ N. Flyer*
- **Paper-Relatedness**: The script also processed relatedness information between papers to create (paper1, related with, paper2) triples and QA pairs. This involved mapping paper titles using their IDs and generating QA pairs like:
  - Triple: (A Formal Look at Dependency Grammars, related with, D-Tree Substitution Grammars)
  - QA Pair: *"Which paper is cited or referenced in the paper titled 'D-Tree Substitution Grammars'? $\rightarrow$ A Formal Look at Dependency Grammars*
- **Author-Venue Relationships**: Using publication venue information, (author, published in, venue) triples and QA pairs were created. A mapping of venue names to unique IDs was done, and QA pairs were formatted as:
  - Triple: (N. Flyer, published in, Acta Numerica)
  - QA Pair: *"In which venue has the author N. Flyer published? $\rightarrow$ Acta Numerica*
- **Author-Field Relationships**: The fields of study for each paper were extracted to create (author, works in field, topic) triples and QA pairs. A mapping of topics to unique IDs was created, and QA pairs were generated like:
  - Triple: (N. Flyer, works in field, Computer Science)
  - QA Pair: *"In which field of study does the author N. Flyer work? $\rightarrow$ Computer Science*
- **Co-Authorship**: Co-authorship information was processed to create (author1, co-authored with, author2) triples and QA pairs. For example:
  - Triple: (Iavarone A., co-authored with, Wei Zhang)
  - QA Pair: *"Which author has co-authored with Iavarone A.? $\rightarrow$ Wei Zhang*
- **Venue-Paper Relationships**: Venue information was used to create (venue, published, paper) triples and QA pairs. For instance:
  - Triple: (IEEE Transactions on Medical Imaging, published, Ridge-based vessel segmentation in color images of the retina)
  - QA Pair: *"Which paper was published in the venue 'IEEE Transactions on Medical Imaging'? $\rightarrow$ Ridge-based vessel segmentation in color images of the retina*

All these triples were combined into a single dataset and saved for training purposes. Additionally, a subset of the QA pairs was randomly selected to create the evaluation set. This set was saved separately and used to assess the performance of the trained model.

A total of 12,105 triples were obtained, encompassing approximately 2,000 unique authors, 1,789 publications, 914 venues, and 22 fields of study. These entities are interrelated (as previously described) to model the semantic space of the entities. The validation set is a sample of 1,216 from the constructed dataset.

5. Results

We used the hit rate metric, which measures the proportion of queries for which the relevant document is among the top k most similar documents retrieved. Specifically, hit rate @ 10 evaluates whether the relevant document is within the top 10 results. Additionally, we employed the metrics provided by the InformationRetrievalEvaluator, which are:

- **cos_sim-Accuracy@5**: The accuracy of retrieving the relevant document within the top 5 results based on cosine similarity.
- **cos_sim-Precision@5**: The precision of the top 5 retrieved documents based on cosine similarity.
- **cos_sim-Recall@5**: The recall of retrieving the relevant document within the top 5 results based on cosine similarity.
- **cos_sim-MRR@10**: The Mean Reciprocal Rank of the top 10 results based on cosine similarity, which measures how far down the rank the first relevant document appears [10].
- **cos_sim-NDCG@10**: The Normalized Discounted Cumulative Gain of the top 10 results based on cosine similarity, which evaluates the ranking quality.
- **cos_sim-MAP@100**: The Mean Average Precision of the top 100 results based on cosine similarity, which evaluates precision across recall levels.

As seen in Table 1, the domain-adapted model significantly outperforms the base model in entity retrieval tasks.

Model	Hit Rate@10	Accuracy@5	Precision@5	Recall@5	MRR@10	NDCG@10	MAP@100
Base Model	0.1	0.0699	0.0139	0.0019	0.0517	0.0139	0.0019
Custom Model	0.94	0.8602	0.1720	0.0476	0.6579	0.1674	0.0367

Table 1: Performance comparison between the base model and the domain-adapted model using various evaluation metrics.

It is important to emphasize that, given the use case for IR systems and the way the training was designed (QA), the model is intended to be used by vectorizing the entities obtained from the database and performing the information retrieval exercise based on measuring the similarity with the query embedding. Therefore, it is crucial that the vector of each entity correctly represents its semantic characteristics within the domain context. For instance, using the adapted model, the embedding of "Kudenko D." has a cosine similarity of 0.17 to the question "Which authors are working in Artificial Intelligence?", while "Susan H. Fisher" has only 0.03. However, for the question "Which authors are working in Biology?", Susan has a similarity score of 0.2 compared to Kudenko's 0.09. This result is not a coincidence, as Kudenko is a computer science and AI researcher at Leibniz University, while Susan is a researcher in medicine and biology at the University of California.

This example demonstrates that the adapted model successfully creates a representation of the entities associated with a person's name that captures additional information related to their area of expertise. This capability is vital for ensuring the effectiveness of the information retrieval system, as it allows for more accurate and contextually relevant results.

Finally, we conducted some tests by vectorizing and comparing concepts outside the specific domain to ensure that the adapted model preserved the base capabilities. The experiments included computing the cosine similarity between vectors that represent various concepts beyond the particular domain.

Comparison	Base Model	Custom Model
Shark and Ocean	0.5232	0.5842
Shark and Strawberry	0.2332	0.3927

Table 2: Dot product comparisons between concepts beyond the particular domain for the base model and the custom domain-adapted model.

The results presented in Table 2 provide two examples demonstrating that the domain-adapted model retains the capacity to differentiate between semantically related and unrelated concepts, thus preserving its original semantic functions.

In conclusion, the custom domain-adapted model demonstrates a significantly improved performance in entity retrieval tasks, while still retaining the capacity to identify semantic relationships in general scenarios.

6. Deployment

First, the model was exported to *ONNX* format for interoperability. ONNX, which stands for *Open Neural Network Exchange*, is an open source format designed to represent machine learning models and was created to facilitate the interchangeability of models between different frameworks and tools [11]. Afterwards, the model was deployed on NVIDIA Inference Server with *TensorRT* optimizations to experiment with FT16 and FT32 precision. Note that the deployed model does not include the tokenization steps.

We ran load tests to confirm the the best settings for the model deployment.

The first test consisted of 1 sentence with 50 tokens each using FT16 and FT32 quantization. The results are shown in Table 3 and 4.

Concurrency	Throughput (infer/sec)	Avg Latency (ms)
1	2.25504	380
2	2.63655	689
3	2.28362	1312
4	2.29563	1740

Table 3: Model performance obtained on a NVIDIA A40 GPU with 24GB of memory using FT16 quantization.

Concurrency	Throughput (infer/sec)	Avg Latency (ms)
1	2.25837	441
2	2.7530	723
3	2.2685	1306
4	2.24074	1764

Table 4: Model performance obtained on a NVIDIA A40 GPU with 24GB of memory using FT32 quantization.

As we can observe, there is a substantial improvement when using quantization and it should be strongly considered for productive environments. Although more rigorous tests are needed since results also fluctuated considerably.

The second test was 5 sentences with 50 tokens each using both FT16 and FT32 quantization. The results are shown in Table 5 and 6.

Concurrency	Throughput (infer/sec)	Avg Latency (ms)
1	0.762781	1308
2	0.62128	3125
3	0.649102	4577
4	0.635894	6119

Table 5: Model performance obtained on a NVIDIA A40 GPU with 24GB of memory using FT16 quantization.

Both tests show that the server had started to throttle up with the amount of requests being processed.

6.1. Sentence Transformer Model

We employed the sentence transformer model *all-MiniLM-L12-v2* as a baseline. Although the model was trained on generalized datasets, it still had a decent performance.

Concurrency	Throughput (infer/sec)	Avg Latency (ms)
1	0.79081	1240
2	0.64117	3139
3	0.631441	4695
4	0.631994	6301

Table 6: Model performance obtained on a NVIDIA A40 GPU with 24GB of memory using FT32 quantization.

6.2. Domain Adapter for Sentence Transformer Model

The training time for the domain adapter was considerably low and achieved a minor improvement compared to the baseline model.

7. RAG Agent with LLM

To demonstrate the practical application of our adapted embedding model, we implemented a Retrieval-Augmented Generation (RAG) agent leveraging Large Language Models (LLMs) for answering questions about the dataset.

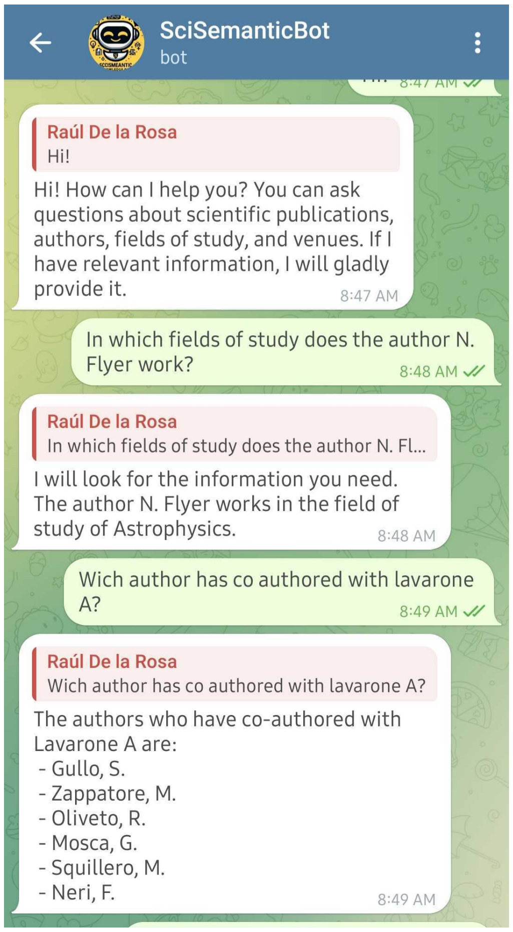
RAG represents a recent advancement in Open-Domain Question Answering (ODQA), seamlessly integrating retrieval-based techniques with generation-based approaches to enhance answer quality and relevance [12], and LLMs are large neural probabilistic models that are typically pretrained on large amounts of data [13].

One of the most popular ways to implement a RAG system is by vectorizing the information that will serve to contextualize the generation of a response and storing it in an optimal format. Once a query is received, it is vectorized, and the top-k information snippets with the highest similarity are retrieved using a metric such as cosine similarity.

For this exercise, all the entities we worked with (authors, venues, fields of study and papers) were vectorized and stored for future retrieval. Given the model's objective according to the implemented training and obtained results, each entity's representation contains information about its multiple relationships with other entities. Thus, once the most similar entities to a query are identified, the associated triples are returned, allowing the LLM to interpret the relationships between these entities and generate a precise response.

The LLM used is Mixtral 8x7b, which offers optimal performance and is provided by the Groq API in its free tier. The model is provided in a highly optimized way to deliver very low response times and with a sufficiently large context window (>32k tokens), making it suitable for our task.

Finally, the agent has been deployed via a Telegram bot named *SciSemanticBot* (@MLT_G4_BOT)4. This was achieved using a Python library that allows running a webhook continuously within a process.



Raúl De la Rosa
Hi!

Hi! How can I help you? You can ask questions about scientific publications, authors, fields of study, and venues. If I have relevant information, I will gladly provide it. 8:47 AM

In which fields of study does the author N. Flyer work? 8:48 AM

Raúl De la Rosa
In which fields of study does the author N. Fl...

I will look for the information you need. The author N. Flyer works in the field of study of Astrophysics. 8:48 AM

Wich author has co authored with lavarone A? 8:49 AM

Raúl De la Rosa
Wich author has co authored with lavarone A?

The authors who have co-authored with Lavarone A are:
- Gullo, S.
- Zappatore, M.
- Oliveto, R.
- Mosca, G.
- Squillero, M.
- Neri, F. 8:49 AM

Figure 4: Example conversation with SciSemanticBot demonstrating various queries about the dataset.

Below are some examples of the entities you can query with SciSemanticBot to test and interact with the model effectively:

- Authors: Kirk Fiedler D., R. Mathies, T. Feder
- Fields of study: Computer Science, Medicine, Engineering
- Venues: Proceedings of the National Academy of Sciences of the United States of America, arXiv.org, Computer Vision and Pattern Recognition

8. Conclusion

In this study, we successfully developed a domain-specific adapter for sentence transformer models to improve semantic search capabilities within network of scientific publications, authors and venues. By fine-tuning the model through a transfer learning approach with a curated dataset of academic publications, we achieved significant performance enhancements, as evidenced by various evaluation metrics. The use of PyTorch and Sentence Transformers allowed for efficient training and deployment of the model, with notable optimizations achieved through NVIDIA Inference Server.

Our implementation of a Retrieval-Augmented Generation (RAG) agent using large language models demonstrated the practical application of the adapted model in real-world scenarios. The agent, accessible via the Telegram bot SciSemanticBot, provides users with accurate and contextually relevant answers to queries about scientific publications included in the dataset. This approach not only enhances information retrieval tasks but also preserves the model's original semantic functions.

The study highlights the importance of domain-specific adaptations for sentence transformer models, particularly in specialized fields where general-purpose models may fall short. Our findings suggest that similar approaches could be applied to other domains, potentially improving the accuracy and relevance of search results across various applications.

References

[1] Massimo. Melucci and Ricardo. Baeza-Yates. *Advanced Topics in Information Retrieval*. The Information Retrieval Series, 33. Springer Berlin Heidelberg, Berlin, Heidelberg, 1st ed. 2011. edition, 2011.

[2] Jinsung Yoon, Sercan O Arik, Yanfei Chen, and Tomas Pfister. Search-adaptor: Embedding customization for information retrieval, 2024.

[3] Uday Kamath, John Liu, and James Whitaker. *Deep Learning for NLP and Speech Recognition*. Springer International Publishing, Cham, 1st ed. 2019. edition, 2019.

[4] Tim Schopf, Dennis N. Schneider, and Florian Matthes. Efficient domain adaptation of sentence embeddings using adapters. In *Proceedings of the Conference Recent Advances in Natural Language Processing - Large Language Models for Natural Language Processings*, RANLP. INCOMA Ltd., Shoumen, BULGARIA, 2023.

[5] Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2018.

[6] Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. Chapman & Hall/CRC Textbooks in Computing. Chapman and Hall/CRC, an imprint of Taylor and Francis, Boca Raton, FL, 1st edition edition, 2009.

[7] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. *ACM Transactions on Database Systems*, 34(3):1–45, 2009.

[8] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

[9] C. Zhao, Ting Hua, Y. Shen, L. Qian, and H. Jin. Automatic mixed-precision quantization search of bert, 2021.

[10] Nick Craswell. Mean reciprocal rank. In Ling LIU and M. Tamer ÖZSU, editors, *Encyclopedia of Database Systems*. Springer, Boston, MA, 2009.

[11] D. Ren, W. Li, T. Ding, L. Wang, Q. Fan, J. Huo, H. Pan, and Y. Gao. Onnxpruner: Onnx-based general model pruning adapter, 2024.

[12] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17, 2023.

[13] T. Adewumi, N. Habib, L. Alkhaled, and E. Barney. On the limitations of large language models (llms): False attribution, 2024.