Domain Adapter for Sentence Transformer Models

Semantic Search

Jhon Stewar Rayo Mosquera

Carlos Raul de La Rosa

Ana Sofía Medina Martínez

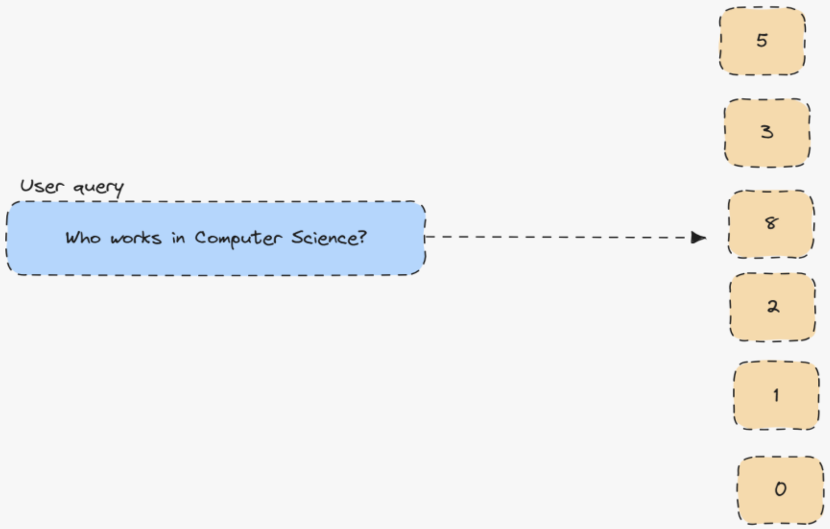
Agenda

- Semantic Search
- Domain Adapters
- Dataset
- Training
- Results
- Deployment

Semantic Search

Traditional search engines use lexical matches to build an index that can be used to quickly return search results to a given query. However, these systems may fail to rank results that are semantically relevant, but do not have exact matches.

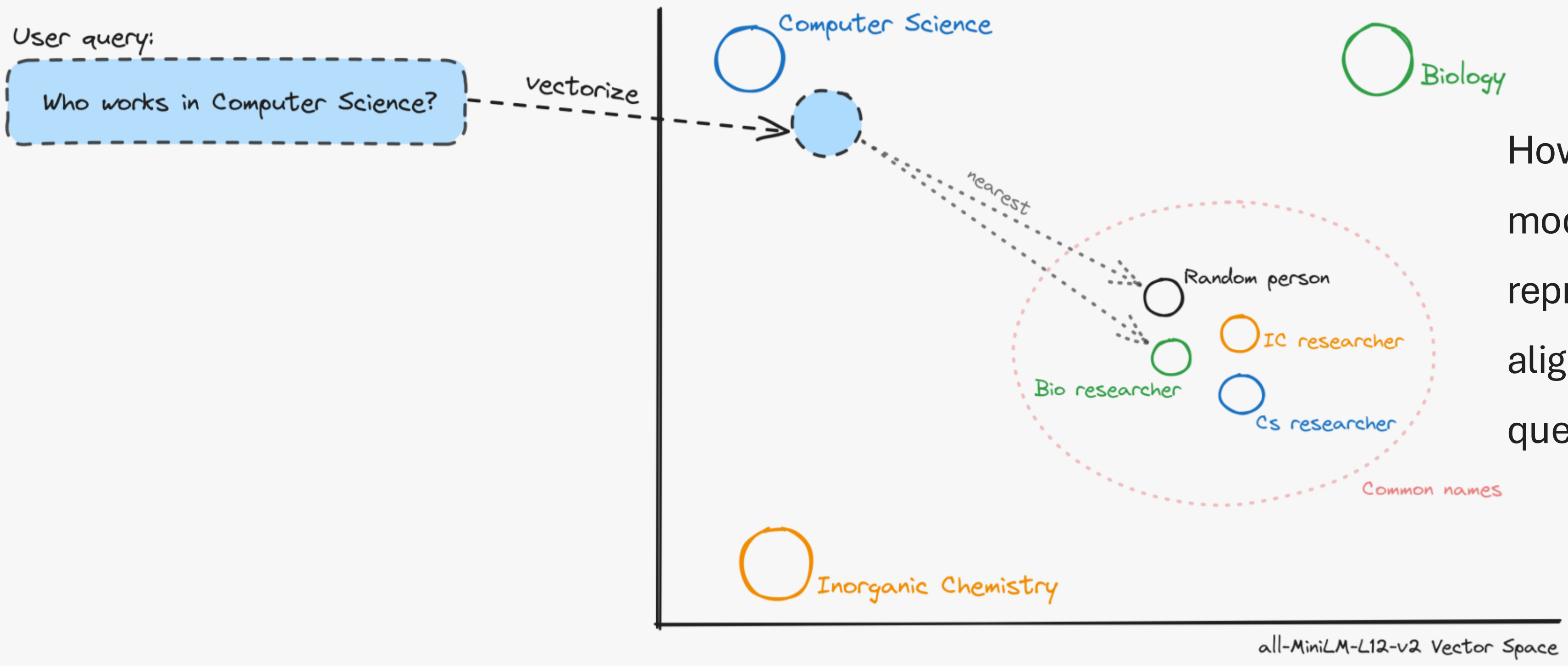
Semantic search aims to solve this problem by using a numeric representation of the corpus and the query.



These representations are usually higher dimensional since each dimension intuitively tries to capture some feature of the input text

Semantic Search

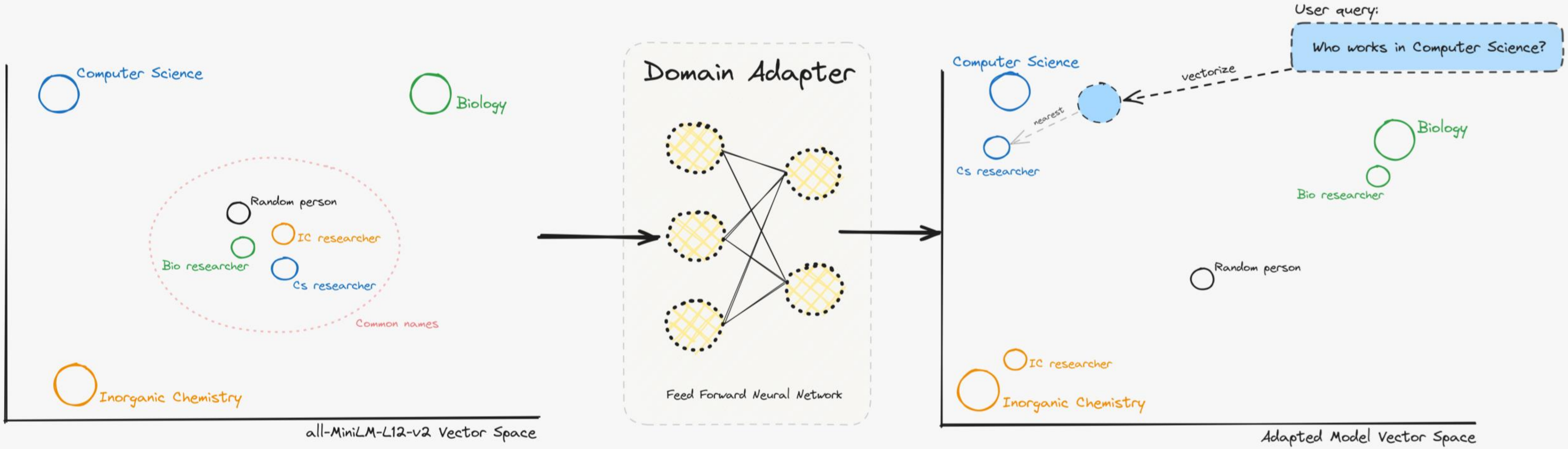
With the rise of language models and the ability to represent text in a vector space (embeddings), the use of semantic search has become one of the most popular strategies for Information Retrieval.



However, when using pre-trained models for general purposes, the representations do not necessarily align with the specific domain of the query.

Domain Adapters

To address this issue, we propose the implementation of a domain adapter based on a feed-forward neural network on top of a sentence embeddings model with all its trainable parameters frozen. This approach transforms the embeddings into a new space that better represents the specific domain.



Domain Adapters

As the base model, we used all-MiniLM-L12-v2, which offers a good balance between performance and efficiency. Given that its weights are frozen, the base model could have been any other; even private ones like OpenAI's ada-002.

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB

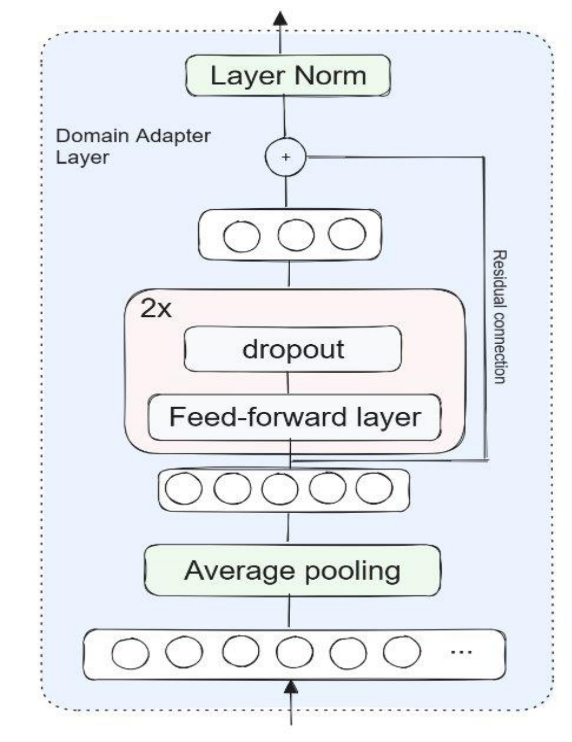
all-MiniLM-L12-v2

Description:	All-round model tuned for many use-cases. Trained on a large and diverse dataset of over 1 billion training pairs.
Base Model:	microsoft/MiniLM-L12-H384-uncased
Max Sequence Length:	256
Dimensions:	384
Normalized Embeddings:	true
Suitable Score Functions:	dot-product (`util.dot_score`), cosine-similarity (`util.cos_sim`), euclidean distance
Size:	120 MB
Pooling:	Mean Pooling
Training Data:	1B+ training pairs. For details, see model card.
Model Card:	https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2

Source: SentenceTransformers documentation

The adapter takes the output of the embeddings model as input and processes it through a proposed architecture based on three dense layers (1024, 512, 384) with ReLU activation function, the last being the output layer with a dimension equivalent to the base model.

Additionally, a dropout of 0.3 was considered along with the addition of a residual factor to minimize the risk of overfitting and to retain the original capabilities of the base model.



Dataset

To tailor the model to the domain of the scientific network encompassing publications, authors, and venues, we opted to utilize the REST API offered by Semantic Scholar. The information retrieved includes references and citations for each publication, as well as details of the authors, journals, areas of study, and more.

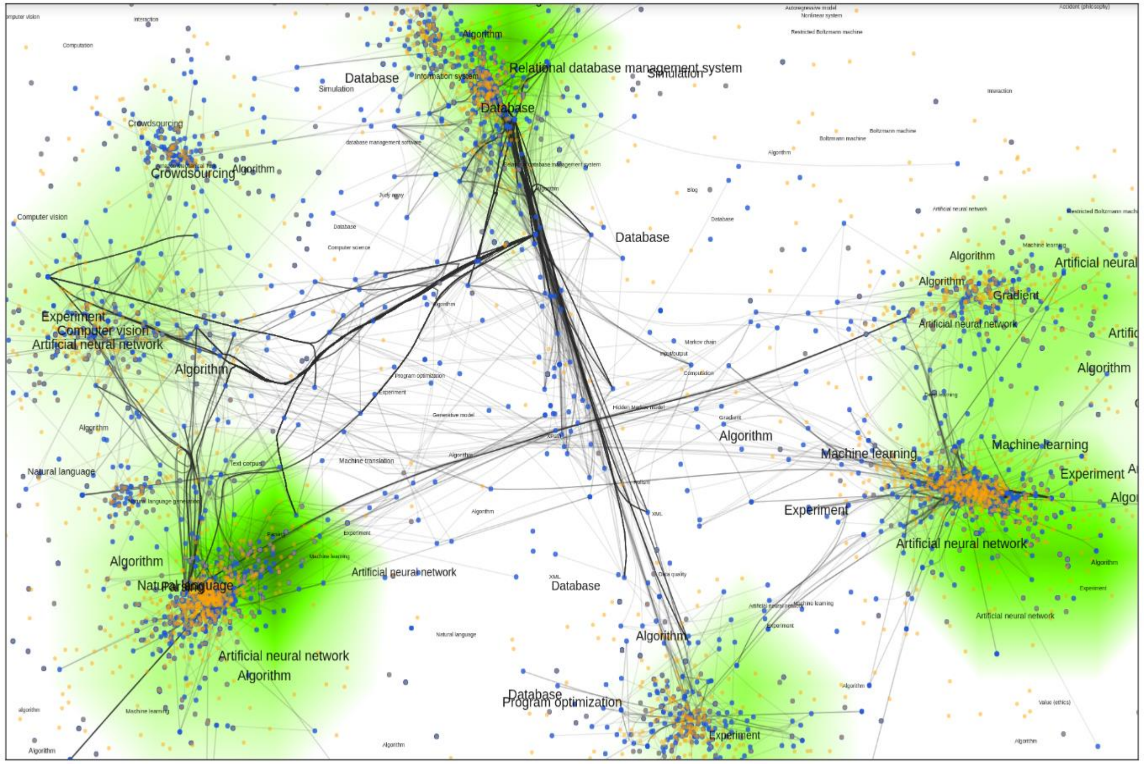


Figure 4: *MODiR* visualisation of Semantic Scholar (S2), all six communities become clear. Authors are blue dots, papers are orange dots, green density map is based on all papers, black opaque edges connect co-authors.

Source: Repke, Tim, and Ralf Krestel. "Visualising Large Document Collections by Jointly Modeling Text and Network Structure." Proceedings of the ACM/IEEE Joint Conference on Digital Libraries. N.p., 2020. 279-288. Web.

We create a total of 12,105 triples of the relationships found for each publication record retrieved and then generate synthetic question-answering (QA) pairs, for example:

- Author-Paper Relationships (author, wrote, paper):
  - Triple: (N. Flyer, wrote, Solving PDEs with radial basis functions)
  - QA Pair: "Who wrote the paper titled 'Solving PDEs with radial basis functions'?" → N. Flyer

This was done to achieve the optimal format for the loss function during training and to ensure correct associations between entities.

Training

The entire architecture implementation was carried out using PyTorch and the SentenceTransformer class provided by the library of the same name. This integration allowed us to streamline our development process and ensure compatibility with state-of-the-art language models.

The training was conducted on an NVIDIA A40 GPU with 24GB of memory using the SentenceTransformer methods for model finetuning. We employed the ***MultipleNegativesSymmetricRankingLoss*** function, which aims to maximize the similarity between positive pairs while minimizing it between negative pairs.

The training process spanned 500 epochs with a batch size of 256.

```python
from sentence_transformers import SentenceTransformer, losses
from sentence_transformers.evaluation import InformationRetrievalEvaluator

# Define the custom Sentence Transformer model that includes the adapter
custom_domain_model = SentenceTransformer(
    modules=[word_embedding_model, pooling_model, adapter, normalize], device=device
)

# Define the loss function
# MultipleNegativesSymmetricRankingLoss is suitable for information retrieval tasks with positive pairs
train_loss = losses.MultipleNegativesSymmetricRankingLoss(custom_domain_model)

# Define the evaluator
# InformationRetrievalEvaluator evaluates the model on a set of queries and corpus
evaluator = InformationRetrievalEvaluator(
    qa_eval['queries'],
    qa_eval['corpus'],
    qa_eval['relevant_docs'],
    name='qa_eval',
    main_score_function='dot_score'
)

# Define the number of epochs and warmup steps
epochs = 500
warmup_steps = int(len(loader) * epochs * 0.1)

# Train the model
# fit() trains the model with the specified training objectives and evaluator
custom_domain_model.fit(
    train_objectives=[(loader, train_loss)],  # Training objectives: DataLoader and loss function
    epochs=epochs,  # Number of epochs
    warmup_steps=warmup_steps,  # Number of warmup steps
    output_path='results/domain_adaptation_model',  # Output path to save the trained model
    show_progress_bar=True,  # Display progress bar during training
    save_best_model=True,  # Save the best model according to evaluation
    use_amp=True,  # Enable Automatic Mixed Precision (AMP)
    evaluator=evaluator,  # Evaluator to assess the model during training
    evaluation_steps=50,  # Evaluate the model every 50 steps
)
```

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. Additionally, we employed the metrics provided by the ***InformationRetrievalEvaluator***. The domain-adapted model significantly outperforms the base model in entity retrieval metrics.

Model	Hit-Rate@10	MRR@10	NDCG@10	MAP@100
Base Model	0.10	0.05	0.01	0.002
Adapted Model	0.94	0.65	0.16	0.036

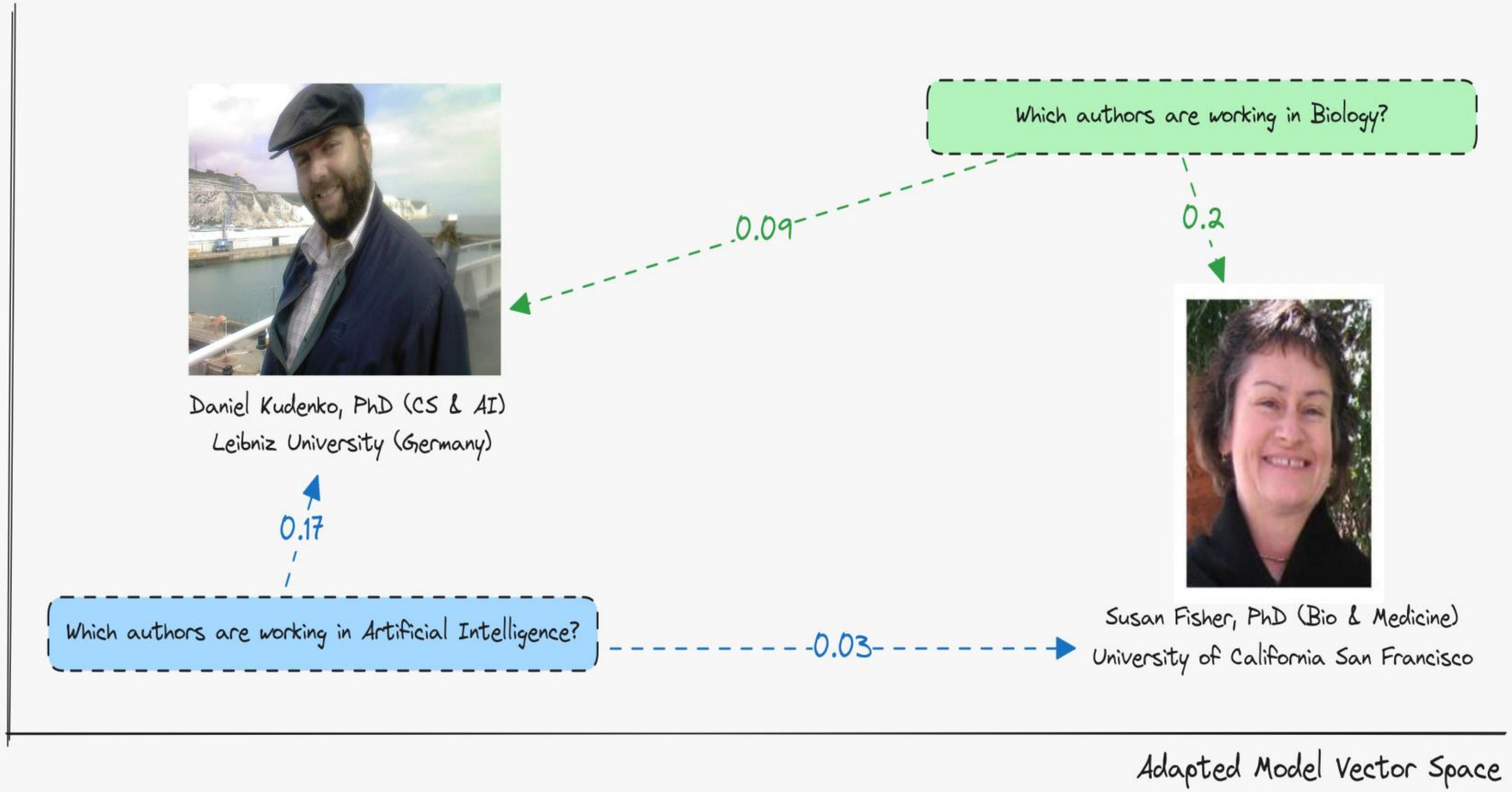
We conducted tests comparing concepts outside the specific domain to ensure the adapted model preserved the base capabilities. The adapted model effectively differentiates between semantically related and unrelated concepts demonstrating that it retains its original semantic functions.

	Dot Product	
Comparison	**Base Model**	**Adapted Model**
Shark & Ocean	0.52	0.58
Shark & Strawberry	0.23	0.39

Results

The adapted model creates representations that accurately capture the semantic characteristics of entities within the domain context. For example, when dealing with entities corresponding to a person's name, the model can generate representations that include information related to their area of expertise.

This capability is crucial for ensuring the effectiveness of the information retrieval system, as it enables more accurate and contextually relevant results.



Deployment

The model was adapted and converted to ONNX format – an open format that enables interoperability – so it can be used in other frameworks.

We demonstrate how the model can be optimized for production using *TensorRT* quantization and measured its latency and throughput on a NVIDIA A40 GPU with 24GB of RAM.

Concurrency	**Throughput (infer/sec)**	**Avg Latency (ms)**
1	2.25	380
2	2.63	689
3	2.28	1312
4	2.29	1740

FT16 quantization

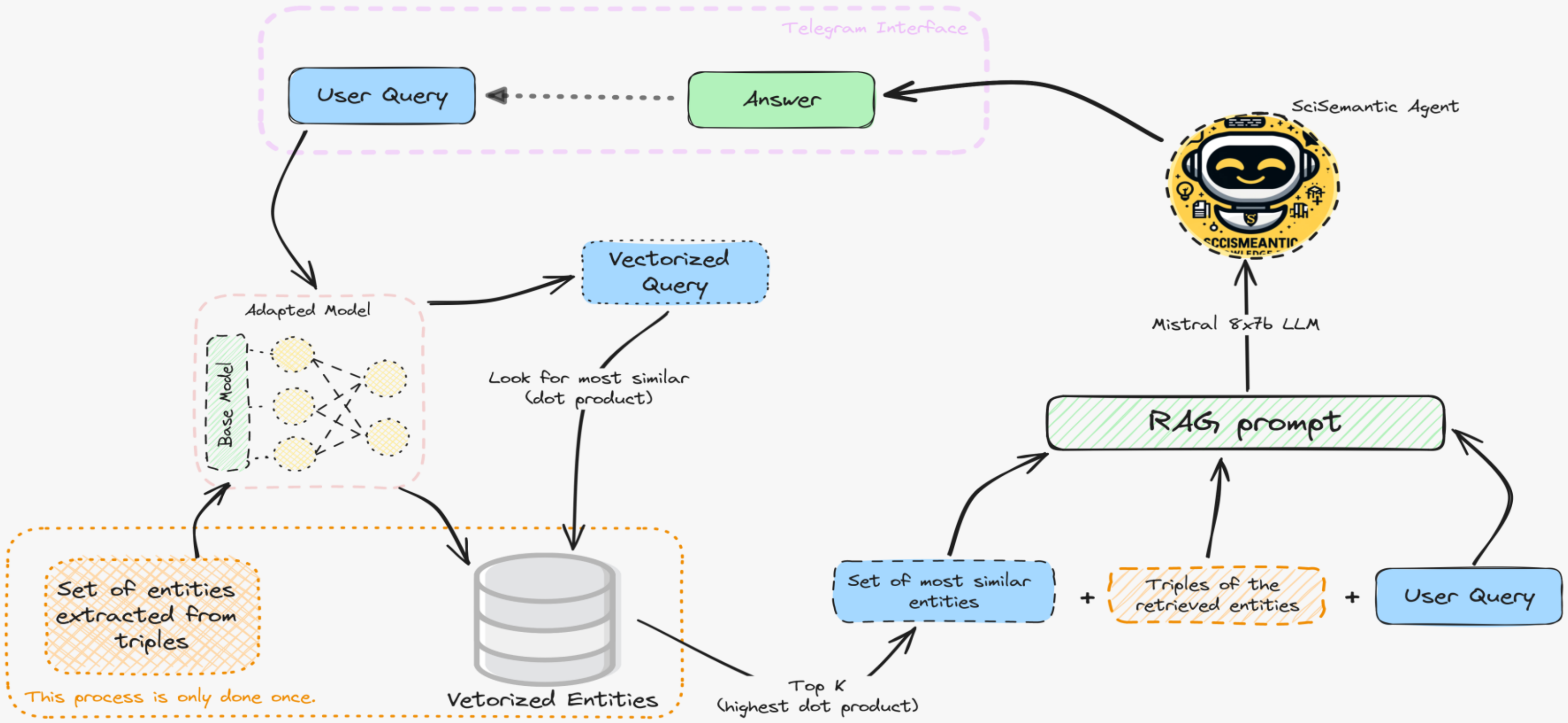
Concurrency	**Throughput (infer/sec)**	**Avg Latency (ms)**
1	2.25	441
2	2.75	723
3	2.26	1306
4	2.24	1765

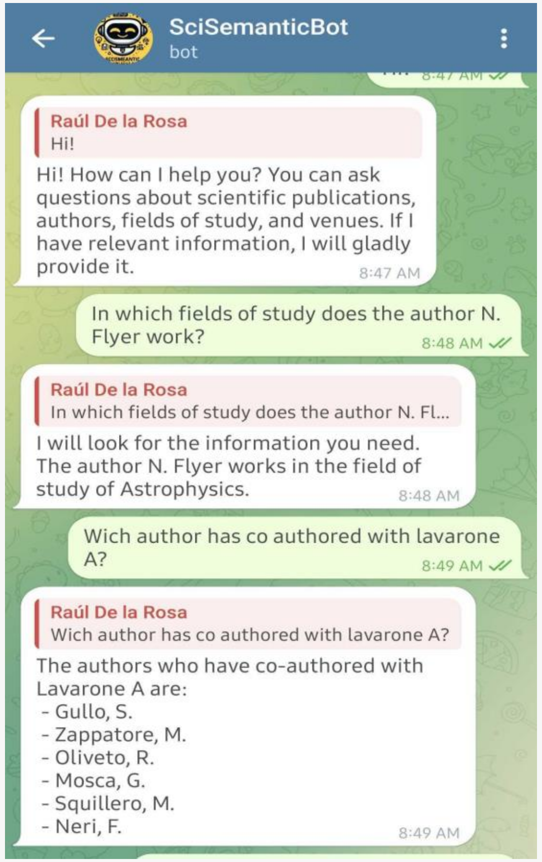
FT32 quantization

Triton Inference Server | NVIDIA Developer

Deployment

To showcase a practical application of our adapted embedding model in a real-world and highly popular use case, we implemented a Retrieval-Augmented Generation (RAG) agent using Large Language Models (LLMs) to answer questions about the dataset. The agent has been deployed via a Telegram bot named SciSemanticAgent (@MLT_G4_BOT).





Conclusions

We successfully developed a domain-specific adapter for sentence transformer models, significantly improving semantic search capabilities within a network of scientific publications, authors, and venues. This study underscores the importance of domain-specific adaptations for sentence embedding models, particularly in specialized fields where general-purpose models may not perform adequately.

Our results suggest that this method could be extended to other domains or applications (e.g., Topic Modeling), potentially enhancing the accuracy and relevance of search results in diverse contexts.

References

- 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

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

- 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