Applying Retrieval-Augmented Generation to Linguistic Research

by

Arif Arabacı

B.A. in Linguistics, Boğaziçi University

Boğaziçi University

2025

ABSTRACT

The growing volume of academic literature in linguistics makes it increasingly challenging for researchers and students to efficiently find and synthesize relevant information. Traditional search methods often yield irrelevant results or cannot capture relevant articles, while reviewing large numbers of abstracts is time-consuming. This thesis explores the application of Retrieval-Augmented Generation (RAG) to address these challenges by integrating dense vector-based retrieval with generative language models.

The system uses dense embeddings and cosine similarity to identify semantically relevant articles, while incorporating metadata such as publication date and popularity to refine retrieval rankings. The most relevant abstracts are then used as context for generating concise summaries or responses to user queries.

Evaluation shows that the system effectively improves retrieval accuracy and reduces the time required for literature review, highlighting its potential as a tool for linguistics researchers. This study demonstrates the implementation of RAG in domain-specific knowledge and offers a solution for literature retrieval and summarization.[1]

[1] https://github.com/arifarabaci/RAG-for-Lingbuzz

1. Problem Statement

The field of linguistics, like many other academic disciplines, is experiencing an exponential growth in the volume of literature published every year. This includes every type of academic work that contributes to the body of knowledge in areas such as syntax, semantics, sociolinguistics, and computational linguistics. This growth creates challenges for students and researchers. Keeping up with the sheer volume of new research is becoming increasingly difficult, even for those who specialize in narrow subfields.

The first and one of the most important steps in the academic research process is finding and evaluating relevant literature. Students, while searching for relevant literature for their thesis, typically search through online libraries by manually reading through abstracts, skimming entire papers, and determining the relevance of each work to their study. This activity can be time-consuming, particularly for researchers who have limited time and deadlines to meet. Moreover, traditional search systems often rely on keyword matching, which can produce inaccurate results when faced with ambiguous terms or specific jargon. The need for a system that not only retrieves relevant research but also answers the researcher's query on the topic is apparent.

Retrieval-Augmented Generation (RAG) might fulfill this need. RAG systems can search for relevant papers and provide summaries of their abstracts or main content, by combining retrieval mechanisms with generative capabilities, reducing the time spent on literature review. This study aims to explore the implementation of a RAG system built to help students and researchers find and evaluate relevant literature efficiently.[2]

2. Introduction to Retrieval-Augmented Generation (RAG)

The advance of large pre-trained language models has revolutionized the field of natural language processing, enabling models to store and access vast amounts of factual knowledge within their parameters. However, this parametric memory approach has its limitations: it is difficult to update or expand, offers limited transparency into how decisions are made, and often generates "hallucinated" content that cannot be easily verified. To address these issues, researchers have

[2] https://github.com/arifarabaci/RAG-for-Lingbuzz

developed hybrid models that combine external and internal resources, offering a more dynamic and interpretable way to access external knowledge.

RAG integrates a retrieval mechanism into the language generation process, allowing the model to fetch relevant documents from a database and generate output on this retrieved information. The RAG framework consists of two main components: a retriever that identifies and ranks relevant text passages based on the input query, and a generator that uses these passages, along with the original query, to produce a coherent and factual response. This approach enhances the model's ability to generate more accurate, and contextually relevant outputs across knowledge-intensive tasks such as question answering, summarization.

3. Methodology

For this study, the dataset was sourced from LingBuzz, a prominent online archive and repository for linguistics research. LingBuzz hosts a wide range of articles, preprints, and publications from various subfields of linguistics, including syntax, semantics, phonology, and computational linguistics. The platform is recognized for its open-access policy, allowing researchers and students to freely access research in the field, which also makes it a perfect source for the system.

The dataset includes articles retrieved from LingBuzz, including their titles, abstracts, authors, publication metadata, and keywords. This rich data provides a solid foundation for training and testing a RAG system aimed at assisting linguistics researchers.To ensure the dataset was suitable for use in a RAG system, several preprocessing steps were applied to enhance the quality and consistency of the data. The preprocessing process focused on preparing the textual content for embedding generation and retrieval, while maintaining the integrity of the original research articles.

First, text normalization was performed on the titles, abstracts, and keywords. This involved cleaning the text to remove special characters, redundant whitespace, and any unnecessary formatting that could interfere with the embedding process. All textual data was converted to lowercase to ensure uniformity and reduce redundancy during similarity calculations.Next, the dataset was examined for missing or incomplete data. Articles without titles or abstracts, or with incomplete entries in these fields, were excluded to preserve the quality of the retrieval process.

Finally, the textual data was encoded into dense vector representations using a pre-trained language model. These embeddings serve as the basis for similarity-based retrieval within the RAG framework. The embeddings capture semantic information from the text, enabling the system to identify and retrieve relevant documents even for complex or specific queries.

3.1. Dense Vector Representation and Embedding Generation

Dense vector representations, or embeddings, are foundational in modern NLP tasks, including retrieval systems. These embeddings are numerical vectors that capture the semantic meaning of text, enabling sophisticated similarity-based retrieval. Unlike traditional text representations like one-hot encoding or bag-of-words, which are sparse and high-dimensional, embeddings are continuous, dense vectors that represent words, sentences, or documents in a fixed-dimensional space (Pinhanez & Cavalin, 2022). This approach allows for capturing deeper semantic relationships, where text with similar meanings is represented by vectors that are close to each other in the embedding space. For example, terms such as "syntax theory" and "syntactic analysis" would have embeddings that are closely aligned due to their shared linguistic context.

To generate these embeddings, this study utilized BERT (Bidirectional Encoder Representations from Transformers), a transformer-based model known for its ability to produce contextualized embeddings. Unlike traditional models that process text in a unidirectional manner, BERT processes text bidirectionally, considering the context of a word from both its preceding and succeeding words (Devlin et al., 2019). This bidirectional understanding allows BERT to generate embeddings that encapsulate both semantic and syntactic nuances of text.

Additionally, Sentence-Transformers were employed to enhance embedding generation, particularly for tasks requiring sentence-level representations. Sentence-Transformers, which extend models like BERT, are fine-tuned on tasks such as Semantic Textual Similarity (STS), optimizing them to produce embeddings that perform well in similarity-based tasks.

The embeddings generated by BERT and Sentence-Transformers were central to the retrieval process within the RAG system. These dense vectors were indexed using a similarity search library, such as FAISS, to enable efficient and accurate retrieval. During retrieval, a query was transformed into its corresponding embedding, which was then compared to the precomputed document embeddings using similarity metrics such as cosine similarity. This process facilitated

the identification of semantically relevant documents, even when the query did not include exact matches with the document text.

Dense vector representations significantly enhanced the quality of retrieval by enabling the system to capture nuanced relationships between queries and documents. Their integration ensured that the RAG system was robust, scalable, and capable of handling the diverse and complex nature of linguistic datasets.

3.2. Hugging-Face Transformers

In addition to using Sentence-Transformers and BERT for embedding generation, this study explored the integration of Hugging Face Transformers to evaluate its potential impact on performance. Hugging Face Transformers is a widely used library that provides a unified interface for numerous pre-trained transformer-based models, enabling tasks such as text classification, generation, and embedding creation. Its flexibility and state-of-the-art models make it a valuable tool for experimentation in retrieval and generation tasks within the RAG framework.

For embedding generation, pre-trained models from Hugging Face's transformers library, such as RagRetriever and RagTokenizer, were utilized (Lewis et al., 2020). These models provide a general-purpose contextual understanding of text, which was adapted for this study's linguistic dataset. Unlike Sentence-Transformers, which are fine-tuned explicitly for tasks like semantic similarity, the models from Hugging Face were used primarily in their pre-trained forms. This approach allowed for a direct comparison of the embeddings' quality and their effectiveness in document retrieval.

These embeddings were subsequently indexed using FAISS for similarity-based retrieval, similar to the setup with Sentence-Transformers. By embedding queries into the same vector space, the retrieval process identified documents with the highest semantic relevance, enabling a fair comparison of performance across different embedding methods.

The use of Hugging Face Transformers also extended to the generation component of the RAG system. Models like facebook/rag-sequence-nq, a pre-built RAG model available through Hugging Face, were integrated to handle both retrieval and generation tasks (Wolf et al., 2019). This model incorporates a retriever and a generator, streamlining the process of providing context-aware outputs based on retrieved documents. The retriever, leveraging FAISS-backed datasets, selected

the most relevant passages, which were then passed to the generator for producing coherent and accurate responses to user queries.

The inclusion of Hugging Face Transformers in the pipeline highlighted its versatility and ease of integration, offering insights into how different transformer-based models impact the overall performance of the RAG system. This exploration contributes to understanding the balance between domain-specific fine-tuning and the broader applicability of pre-trained models in linguistic research.

3.3. Retrieval Mechanism

The retrieval component of the system is a critical part of the RAG framework, responsible for identifying the most relevant documents based on user queries. This system employs a hybrid approach, combining dense vector similarity with metadata-based scoring to refine the retrieval process and align it with the specific needs of researchers.

3.3.1. Cosine Similarity for Dense Vector Matching

To identify semantically similar documents, the system uses cosine similarity as the primary metric to compare the dense vector representations of the query and the documents. Cosine similarity measures the cosine of the angle between two vectors in a multidimensional space, ranging from -1 (completely dissimilar) to 1 (identical). Dense vectors for the documents and queries are generated using Sentence-Transformers or Hugging Face Transformers, as discussed in the embedding section.

For a given query, its embedding is computed and compared with the pre-computed embeddings of all documents in the dataset. The similarity scores are then used to rank the documents, prioritizing those that are most relevant to the query in semantic space. This process ensures that the retrieval system captures context and meaning beyond simple keyword matching.

3.3.2. Metadata-Based Scoring

To further refine the retrieval process, the system incorporates metadata from the dataset. This metadata includes fields such as upload date and download counts, which provide additional context about the relevance and popularity of the articles. Depending on the nature of the query, specific metadata fields are weighted to adjust the overall relevance score:

- **Recency**: For queries emphasizing recent research, such as *"What are the latest developments in syntax theory?"*, articles published more recently are given a boost in their score. A fixed weight is added to the cosine similarity score for recent articles.
- **Popularity**: For queries focusing on widely-read or influential works, such as *"Most popular papers on phonology"*, articles with higher download counts are prioritized by adding a proportional score based on their popularity.

3.3.3 Hybrid Scoring System

The final score for each article is a combination of the cosine similarity score and metadata-based adjustments. The scoring formula can be expressed as:

$$\text{Final Score} = \text{Cosine Similarity} + \sum(\text{Metadata Weights})$$

For example, if an article matches closely with the query based on cosine similarity (e.g., 0.75) and is recent (adding 0.25) and highly downloaded (adding 0.10), its final score becomes 1.10.

This hybrid approach ensures that the system balances semantic relevance with contextual factors like recency and popularity, making the retrieval process more adaptable to different types of user queries.

3.4. Generation

The generation component of the system is responsible for producing coherent and contextually relevant responses to user queries based on the information retrieved by the retriever. In this study, the generation process leverages 'Cohere' language model (command-r-plus-04-2024) to transform the retrieved research abstracts into a synthesized answer or summary (Cohere, 2024).

This component was implemented with a relatively straightforward prompt engineering approach. The top-K abstracts retrieved by the system were concatenated to form the context, which was then passed to the model along with the query. The prompt structure was designed to guide the model toward generating a focused and accurate response.

The focus of the study was not on the generation component, but rather on the retrieval mechanism and its integration with metadata-based scoring. As such, minimal effort was dedicated to optimizing the generative process or fine-tuning the model for the specific dataset. Instead, the

goal was to use the generative model in its pre-trained state to evaluate its effectiveness when combined with the retrieval component.

Despite the straightforward approach, using a pre-trained model like Cohere's command-r-plus-04-2024 provides significant advantages. The model is designed for high-quality text generation, capable of producing coherent and contextually aware outputs even without extensive customization. By passing the abstracts as context, the generation process benefits from domain-specific information without requiring the model itself to be fine-tuned on linguistics-specific data.

The simplicity of the generation implementation serves as a baseline for the system, allowing for future extensions or optimizations if desired. Potential enhancements, such as fine-tuning the model for linguistics-related tasks or experimenting with alternative prompting strategies, are discussed as future directions.

In summary, the generation component, while crucial for completing the RAG pipeline, was intentionally designed with a minimalistic and general-purpose approach. This decision reflects the primary focus of the study on the retrieval mechanism, while demonstrating that even with basic integration, the generative model can effectively synthesize information from retrieved abstracts to address user queries.

4. Practical Work Flow

1. A query is input by the user, such as *"Recent advances in computational linguistics"*.
2. The query is embedded into the same vector space as the documents using the embedding model.
3. Cosine similarity scores are computed between the query embedding and document embeddings using FAISS.
4. Metadata fields (e.g., upload date and download counts) are evaluated based on the query's intent.
5. The cosine similarity scores and metadata-based adjustments are combined to produce the final ranking.

6. The top ranked documents are retrieved and passed to the generation component of the RAG pipeline for summarization or answer generation.

5. Conclusion & Discussion

This study demonstrated the implementation of RAG for improving literature retrieval and summarization for linguistics research. By integrating dense vector similarity with metadata-based scoring, the system achieved enhanced retrieval relevance and context-aware generation. This system can serve as an academic assistant for researchers, enabling efficient identification of relevant works and fostering more comprehensive literature reviews. Moreover, it can be implemented to any e-library to enhance searching and filtering in the future. In spite of its strengths, the system's reliance on pre-trained models may introduce biases, and its performance is constrained by the range of the dataset used for training and evaluation. Future work could focus on fine-tuning for specific subfield, expanding the dataset to include multilingual corpora, and integrating advanced metadata analysis to enhance retrieval performance.

References

1. Cohere. (2024). *Cohere language models*. https://cohere.com
2. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P., Lomeli, M., Hosseini, L., Jégou, H., & Szilvasy, G. (2024). The Faiss library. *arXiv*. https://arxiv.org/abs/2401.08281
3. Harris, C.R., Millman, K.J., van der Walt, S.J. et al. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2
4. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense passage retrieval for open-domain question answering. *arXiv*. https://arxiv.org/abs/2004.04906
5. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv*. https://arxiv.org/abs/2005.11401
6. pandas development team. (2020). *pandas-dev/pandas: Pandas*. Zenodo. https://doi.org/10.5281/zenodo.3509134
7. Pinhanez, C., & Cavalin, P. (2022). Exploring the advantages of dense-vector to one-hot encoding of intent classes in out-of-scope detection tasks. *arXiv*. https://arxiv.org/abs/2205.09021
8. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *arXiv*. https://arxiv.org/abs/1908.10084
9. Starke, M. LingBuzz. https://lingbuzz.net
10. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A. M. (2019). HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv*. https://arxiv.org/abs/1910.03771

Github Link: https://github.com/arifarabaci/RAG-for-Lingbuzz