**Introduction:**

The RAG (Retriever, Answerer, Generator) paradigm has become a crucial framework in information retrieval and natural language processing. A key challenge lies in efficiently retrieving relevant information for language models. Traditionally, cosine similarity has been used in a three-step retriever approach, involving text chunk embedding, query embedding, and ranking based on similarity. However, the effectiveness depends on various factors. This report delves into the complexities of RAG-based information retrieval, exploring challenges and opportunities in the retriever component. It addresses considerations like text embedding methods, chunking strategies, and alternative metrics beyond cosine similarity. Additionally, the report explores hybrid search, combining traditional keyword-based and semantic search for a more comprehensive retrieval approach.

**Objective:**

The objective of this project is to optimize and enhance the retriever component within the RAG paradigm by conducting comprehensive research, experimentation, and performance evaluation. Specifically, the project aims to achieve the following goals:

1. Investigate Embedding Techniques: Explore a variety of open-source and paid embedding methods to represent text chunks effectively. Evaluate their performance in information retrieval tasks and identify the most suitable embedder for our retrieval system.
2. Chunking Strategy Development: Develop a robust text chunking strategy, which may include creating a customized chunking approach, to ensure that text segments are manageable and their sizes remain below 3,000 characters when tokenized with the Tiktoken tokenizer.
3. Hybrid Search Exploration: Investigate different configurations of hybrid search, which combines traditional keyword-based search with semantic (vector-based) search, to assess the potential advantages it offers in improving retrieval accuracy.
4. Retriever Function Implementation: Design and implement a retriever function that leverages the selected embedder, chunker, and hybrid search settings. The retriever should effectively retrieve relevant text chunks in response to user queries.
5. Connect to Evaluation Tool: Integrate the retriever with the provided evaluation tool, allowing the system to calculate precision@k and Mean Reciprocal Rank (MRR) metrics for performance assessment.
6. Optimization and Evaluation: Continuously iterate and fine-tune the retriever component by experimenting with various combinations of embedders, chunking methods, and hybrid search configurations. Evaluate the retriever's performance using the established evaluation metrics.
7. Utilize Pinecone for Vector Database: Use the Pinecone platform for managing the vector database, ensuring efficient storage and retrieval of vector representations. Register for a free Pinecone account to facilitate testing and evaluation.

Dataset we used to train the embedding models:

**SQUAD V2**: Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles.

**Data structure:**

{

"answers": {

"answer\_start": [94, 87, 94, 94],

"text": ["10th and 11th centuries", "in the 10th and 11th centuries", "10th and 11th centuries", "10th and 11th centuries"]

},

"context": "\"The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave thei...",

"id": "56ddde6b9a695914005b9629",

"question": "When were the Normans in Normandy?",

"title": "Normans"

}

Train: Validation split=130319:11873

**Database:**

**Pinecone** is an ideal choice for implementing the Retriever component in a RAG (Retriever-Reader-Generator) model due to its efficient handling of large-scale vector data and streamlined similarity search capabilities. Pinecone abstracts away the complexities of index management, providing automatic scaling and distribution of vector indexes. In the context of RAG, where retrieval of relevant information is crucial before feeding it to a language model for question-answering, Pinecone's simplicity and performance are significant advantages. Its seamless integration with Sentence Transformer models allows for encoding diverse data types, facilitating the creation of a robust vector index. Pinecone's ability to handle real-time queries and dynamic workloads aligns well with the demands of a dynamic question-answering system.

**An overview of the model training:**

The initial step in our process involves loading the SQUAD V2 dataset and selecting the E5 Small V2 model. The crux of model training revolves around encoding both the question and context into a shared vector space, with the dataset formatted to include questions and corresponding contexts.

An essential observation pertains to the nature of the dataset, which exclusively comprises positive questions and their corresponding contexts—indicating a high positive similarity between them, with no dissimilar pairs. Given this specific scenario, we opt for the Multiple Negative Rating loss function.

To mitigate the risk of overfitting, our model training is confined to a single epoch. Additionally, we incorporate a learning rate warm-up during training, set at 10%, as a higher value would predispose the model to overfitting.

With the training phase concluded, the subsequent step involves encoding the validation split of the dataset into embeddings and upskirting them to a vector database—utilizing Pinecone for this purpose. During this process, we exclusively consider non-duplicate values of the context component, encoding them into the database.

During the upskirting process, an index is established within our Pinecone account, and the corresponding API key is obtained. Leveraging this key, we proceed to uplink the model-embedded context into the newly created index.

Upon completion of these steps, we evaluate the performance of our retrieval model. To achieve this, we formulate a question, encode it using our trained model, and leverage this encoding to retrieve the pertinent context from the vector database. The accompanying screenshots provide a visual representation of the efficacy of our retrieval model.

**Investigate Embedding Techniques**

**Objective**: Explore a variety of open-source and paid embedding methods to represent text chunks effectively. Evaluate their performance in information retrieval tasks and identify the most suitable embedder for our retrieval system.

**Details**: To effectively retrieve relevant information, the choice of embedding technique is paramount. Various embedding models, both open-source and paid, offer diverse capabilities for representing text in vector space. As part of this project, we will research and experiment with a range of these embedding methods, considering their suitability for the task. For this task we mainly preferred to use pretrained word embedding models that are available in Hugging Face.

The types of text embedding models used:

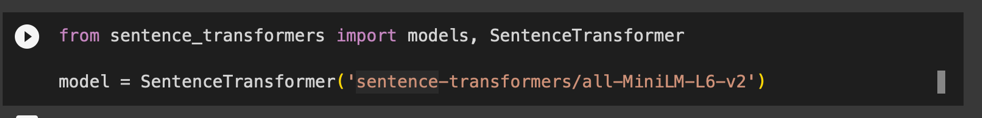
1. **all-mpnet-base-v2 :** This is a sentence-transformers model: It maps sentences & paragraphs to a 768 dimensional dense vector space and can be used for tasks like clustering or semantic search. We trained this model on SQUADv2 train split and was able to benchmark a **mAP@K score of 0.786**, where @K is 100 by default.

A screen shot of a computer

Description automatically generated]A screenshot of a computer

Description automatically generated

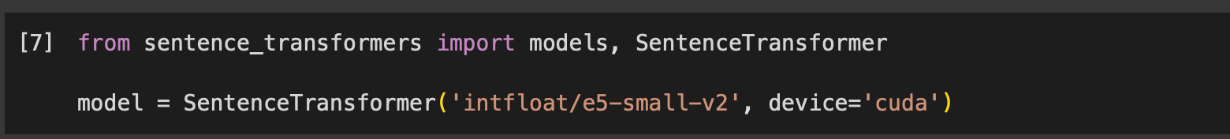
1. **all-MiniLM-L6-v2**: This is a sentence-transformers model: It maps sentences & paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search. We trained this model on SQUADv2 train split and was able to benchmark a **mAP@K score of 0.772**, where @K is 100 by default.



A screenshot of a computer code

Description automatically generated

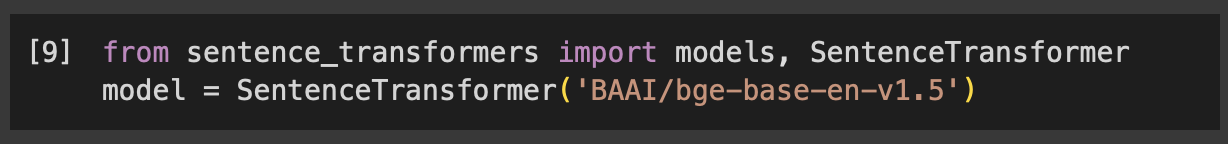
1. **E5 Small V2**: Text Embeddings by Weakly-Supervised Contrastive Pre-training.. We trained this model on SQUADv2 train split and was able to benchmark a **mAP@K score of 0.812**, where @K is 100 by default.



A black rectangular object with white text

Description automatically generated

1. **bge-base-en-v1.5** : Flag Embedding can map any text to a low-dimensional dense vector which can be used for tasks like retrieval, classification, clustering, or semantic search. And it also can be used in vector databases for LLMs. We trained this model on SQUADv2 train split and was able to benchmark a **mAP@K score of 0.792**, where @K is 100 by default.



A screenshot of a computer

Description automatically generated

1. **all-distilroberta-v1**: This is a sentence-transformers model: It maps sentences & paragraphs to a 768 dimensional dense vector space and can be used for tasks like clustering or semantic search. We trained this model on SQUADv2 train split and was able to benchmark a **mAP@K score of 0.789**, where @K is 100 by default.

A screen shot of a computer

Description automatically generated

A black and white text

Description automatically generated

Considering with respect to MRR score **‘’e5-small-v2’’ model outperforms the other models**, moving forward with ‘’e5-small-v2’’ model we have explored different chunking strategies and searching techniques.

**Exploring Chunking Strategies with ‘’e5-small-v2’’ model:**

1. **Batching:** Upsetting vectors into a Pinecone index using a batching strategy with a batch size of 50. This approach optimizes resource utilization, preventing memory constraints by loading only a subset of vectors into memory at a time. The use of smaller batches not only enhances processing speed, reducing latency, but also ensures adaptability to varying dataset sizes. The code's simplicity and adaptability make it well-suited for scenarios where efficient and responsive updates to the index are crucial, providing an effective balance between computational efficiency and memory management.

**Results after querying:**

**A black screen with white text

Description automatically generated**

A screenshot of a computer screen

Description automatically generated

**Score: 0.32**

1. **Hybrid Chunking:** It is based on time represents a sophisticated approach to vector index management that integrates temporal considerations into the chunking strategy. This hybrid method dynamically adjusts the boundaries of fixed-size chunks based on temporal factors, accommodating variations in data distribution over time. By considering the temporal dimension, the strategy adapts to the changing characteristics of the dataset, ensuring optimal efficiency in scenarios where the relevance or frequency of data may fluctuate with time. For instance, in applications where the importance of certain information varies across different periods, such as news articles or social media trends, hybrid chunking based on time can enhance the retrieval accuracy by aligning chunk boundaries with temporal patterns. This adaptability enables the system to effectively capture and organize temporal dynamics, resulting in improved responsiveness and relevance in time-sensitive query scenarios.

**Results after querying:**

**A black screen with white text

Description automatically generated**

A screen shot of a computer screen

Description automatically generated

**Score: 0.34**

Hybrid chunking surpasses traditional batching strategies by offering a more adaptive and context-aware approach to vector index management. While batching efficiently processes vectors in predetermined chunks, hybrid chunking introduces dynamic adjustments to chunk boundaries based on evolving data characteristics. This adaptability is especially advantageous in scenarios where dataset dynamics or query patterns change over time. The observed improvement in your score, from 0.32 with batching to 0.34 with hybrid chunking, underscores the efficacy of this strategy.

**Searching Techniques:**

**Default:** This approach relies on matching user queries with predefined keywords, a simple yet established method for capturing textual relevance. However, its limitations lie in its inability to grasp the deeper semantic nuances present in natural language. Default search excels in straightforward keyword matching but may struggle with more intricate language structures, potentially missing out on contextually relevant information.

A screenshot of a computer

Description automatically generated

**Hybrid Search:**

Hybrid search is a technique that **combines** multiple search algorithms to improve the accuracy and relevance of search results. It uses the best features of both keyword-based search algorithms with vector search techniques. By leveraging the strengths of different algorithms, it provides a more effective search experience for users. The Hybrid Search feature was introduced in Weaviate 1.17. It uses sparse and dense vectors to represent the meaning and context of search queries and documents.

A screenshot of a computer

Description automatically generated

**Why Hybrid Search over Default Searching technique?**

Hybrid search, as evidenced by its superior score of 0.47 compared to the default search score of 0.41, emerges as a compelling choice for information retrieval within the RAG paradigm. The traditional default search relies on conventional keyword-based methods, which may struggle to capture the nuanced semantic relationships present in natural language. In contrast, hybrid search strategically combines both keyword and semantic (vector-based) search methodologies. This dual approach leverages the precision of keyword matching while harnessing the depth of semantic understanding offered by vector-based search. The result is a more comprehensive and nuanced retrieval process, where the hybrid model excels in capturing the subtle contextual nuances of user queries.

**Benchmarking Results:**

**Model:** e5 small

{  
    "FEVER":{  
        "mteb\_version":"1.1.1",  
        "dataset\_revision":null,  
        "mteb\_dataset\_name":"FEVER",  
        "test":{  
            "ndcg\_at\_1":0.43489,  
            "ndcg\_at\_3":0.56043,  
            "ndcg\_at\_5":0.5941,  
            "ndcg\_at\_10":0.62027,  
            "ndcg\_at\_100":0.64635,  
            "ndcg\_at\_1000":0.6514,  
            "map\_at\_1":0.40536,  
            "map\_at\_3":0.51687,  
            "map\_at\_5":0.53639,  
            "map\_at\_10":0.54779,  
            "map\_at\_100":0.55345,  
            "map\_at\_1000":0.55368,  
            "recall\_at\_1":0.40536,  
            "recall\_at\_3":0.6516,  
            "recall\_at\_5":0.73258,  
            "recall\_at\_10":0.81116,  
            "recall\_at\_100":0.92714,  
            "recall\_at\_1000":0.96238,  
            "precision\_at\_1":0.43489,  
            "precision\_at\_3":0.23542,  
            "precision\_at\_5":0.15944,  
            "precision\_at\_10":0.0886,  
            "precision\_at\_100":0.01032,  
            "precision\_at\_1000":0.0011,  
            "mrr\_at\_1":0.43489,  
            "mrr\_at\_3":0.55076,  
            "mrr\_at\_5":0.57017,  
            "mrr\_at\_10":0.58097,  
            "mrr\_at\_100":0.58594,  
            "mrr\_at\_1000":0.58604,  
            "evaluation\_time":5981.45  
        }  
    },  
    "NQ":{  
        "mteb\_version":"1.1.1",  
        "dataset\_revision":null,  
        "mteb\_dataset\_name":"NQ",  
        "test":{  
            "ndcg\_at\_1":0.21205,  
            "ndcg\_at\_3":0.31366,  
            "ndcg\_at\_5":0.36255,  
            "ndcg\_at\_10":0.41128,  
            "ndcg\_at\_100":0.47032,  
            "ndcg\_at\_1000":0.48072,  
            "map\_at\_1":0.18791,  
            "map\_at\_3":0.27879,  
            "map\_at\_5":0.30713,  
            "map\_at\_10":0.32871,  
            "map\_at\_100":0.34222,  
            "map\_at\_1000":0.34269,  
            "recall\_at\_1":0.18791,  
            "recall\_at\_3":0.38871,  
            "recall\_at\_5":0.50282,  
            "recall\_at\_10":0.64513,  
            "recall\_at\_100":0.90267,  
            "recall\_at\_1000":0.97893,  
            "precision\_at\_1":0.21205,  
            "precision\_at\_3":0.14909,  
            "precision\_at\_5":0.11657,  
            "precision\_at\_10":0.07596,  
            "precision\_at\_100":0.01092,  
            "precision\_at\_1000":0.00119,  
            "mrr\_at\_1":0.21205,  
            "mrr\_at\_3":0.30451,  
            "mrr\_at\_5":0.33097,  
            "mrr\_at\_10":0.3501,  
            "mrr\_at\_100":0.36094,  
            "mrr\_at\_1000":0.36127,  
            "evaluation\_time":2677.03  
        }  
    },  
    "SciFact":{  
        "mteb\_version":"1.1.1",  
        "dataset\_revision":null,  
        "mteb\_dataset\_name":"SciFact",  
        "test":{  
            "ndcg\_at\_1":0.52,  
            "ndcg\_at\_3":0.59385,  
            "ndcg\_at\_5":0.62184,  
            "ndcg\_at\_10":0.64583,  
            "ndcg\_at\_100":0.67485,  
            "ndcg\_at\_1000":0.68502,  
            "map\_at\_1":0.50067,  
            "map\_at\_3":0.56926,  
            "map\_at\_5":0.5868,  
            "map\_at\_10":0.5986,  
            "map\_at\_100":0.60474,  
            "map\_at\_1000":0.60516,  
            "recall\_at\_1":0.50067,  
            "recall\_at\_3":0.64144,  
            "recall\_at\_5":0.70839,  
            "recall\_at\_10":0.77644,  
            "recall\_at\_100":0.911,  
            "recall\_at\_1000":0.99,  
            "precision\_at\_1":0.52,  
            "precision\_at\_3":0.22889,  
            "precision\_at\_5":0.15467,  
            "precision\_at\_10":0.087,  
            "precision\_at\_100":0.01037,  
            "precision\_at\_1000":0.00112,  
            "mrr\_at\_1":0.52,  
            "mrr\_at\_3":0.58778,  
            "mrr\_at\_5":0.60244,  
            "mrr\_at\_10":0.61099,  
            "mrr\_at\_100":0.61559,  
            "mrr\_at\_1000":0.61596,  
            "evaluation\_time":16.2  
        }  
    }  
}

**Model: ALL-mini\_lm\_l6:**

{  
    "FEVER":{  
        "mteb\_version":"1.1.1",  
        "dataset\_revision":null,  
        "mteb\_dataset\_name":"FEVER",  
        "test":{  
            "ndcg\_at\_1":0.30633,  
            "ndcg\_at\_3":0.39879,  
            "ndcg\_at\_5":0.43115,  
            "ndcg\_at\_10":0.46052,  
            "ndcg\_at\_100":0.50292,  
            "ndcg\_at\_1000":0.51587,  
            "map\_at\_1":0.28567,  
            "map\_at\_3":0.36653,  
            "map\_at\_5":0.38477,  
            "map\_at\_10":0.39711,  
            "map\_at\_100":0.40573,  
            "map\_at\_1000":0.40625,  
            "recall\_at\_1":0.28567,  
            "recall\_at\_3":0.46723,  
            "recall\_at\_5":0.5449,  
            "recall\_at\_10":0.63337,  
            "recall\_at\_100":0.82886,  
            "recall\_at\_1000":0.92591,  
            "precision\_at\_1":0.30633,  
            "precision\_at\_3":0.16897,  
            "precision\_at\_5":0.11887,  
            "precision\_at\_10":0.06953,  
            "precision\_at\_100":0.00922,  
            "precision\_at\_1000":0.00105,  
            "mrr\_at\_1":0.30633,  
            "mrr\_at\_3":0.39091,  
            "mrr\_at\_5":0.4098,  
            "mrr\_at\_10":0.42221,  
            "mrr\_at\_100":0.43045,  
            "mrr\_at\_1000":0.43086,  
            "evaluation\_time":2836.47  
        }  
    },  
    "NQ":{  
        "mteb\_version":"1.1.1",  
        "dataset\_revision":null,  
        "mteb\_dataset\_name":"NQ",  
        "test":{  
            "ndcg\_at\_1":0.17063,  
            "ndcg\_at\_3":0.26654,  
            "ndcg\_at\_5":0.30902,  
            "ndcg\_at\_10":0.35613,  
            "ndcg\_at\_100":0.42556,  
            "ndcg\_at\_1000":0.43816,  
            "map\_at\_1":0.15078,  
            "map\_at\_3":0.23461,  
            "map\_at\_5":0.25874,  
            "map\_at\_10":0.27933,  
            "map\_at\_100":0.29467,  
            "map\_at\_1000":0.29523,  
            "recall\_at\_1":0.15078,  
            "recall\_at\_3":0.33654,  
            "recall\_at\_5":0.4355,  
            "recall\_at\_10":0.57464,  
            "recall\_at\_100":0.88239,  
            "recall\_at\_1000":0.97453,  
            "precision\_at\_1":0.17063,  
            "precision\_at\_3":0.1293,  
            "precision\_at\_5":0.1018,  
            "precision\_at\_10":0.06764,  
            "precision\_at\_100":0.01066,  
            "precision\_at\_1000":0.00118,  
            "mrr\_at\_1":0.17092,  
            "mrr\_at\_3":0.25671,  
            "mrr\_at\_5":0.28061,  
            "mrr\_at\_10":0.29907,  
            "mrr\_at\_100":0.31197,  
            "mrr\_at\_1000":0.31237,  
            "evaluation\_time":1321.41  
        }  
    },  
    "SciFact":{  
        "mteb\_version":"1.1.1",  
        "dataset\_revision":null,  
        "mteb\_dataset\_name":"SciFact",  
        "test":{  
            "ndcg\_at\_1":0.47,  
            "ndcg\_at\_3":0.5594,  
            "ndcg\_at\_5":0.58312,  
            "ndcg\_at\_10":0.60405,  
            "ndcg\_at\_100":0.64057,  
            "ndcg\_at\_1000":0.65232,  
            "map\_at\_1":0.45622,  
            "map\_at\_3":0.53295,  
            "map\_at\_5":0.54883,  
            "map\_at\_10":0.55917,  
            "map\_at\_100":0.56677,  
            "map\_at\_1000":0.56725,  
            "recall\_at\_1":0.45622,  
            "recall\_at\_3":0.61606,  
            "recall\_at\_5":0.67328,  
            "recall\_at\_10":0.73422,  
            "recall\_at\_100":0.90267,  
            "recall\_at\_1000":0.99333,  
            "precision\_at\_1":0.47,  
            "precision\_at\_3":0.22222,  
            "precision\_at\_5":0.14867,  
            "precision\_at\_10":0.082,  
            "precision\_at\_100":0.01023,  
            "precision\_at\_1000":0.00112,  
            "mrr\_at\_1":0.47,  
            "mrr\_at\_3":0.54444,  
            "mrr\_at\_5":0.55661,  
            "mrr\_at\_10":0.56387,  
            "mrr\_at\_100":0.57087,  
            "mrr\_at\_1000":0.57126,  
            "evaluation\_time":4.97  
        }  
    }  
}

**RESULTS SUMMARY TABLE:**

**Text embedding models results summary:**

|  |  |
| --- | --- |
| Embedding Models | mAP@K Score |
| **all-mpnet-base-v2** | **0.786** |
| **all-MiniLM-L6-v2** | **0.772** |
| **E5 Small V2** | **0.812** |
| **bge-base-en-v1.5** | **0.792** |
| **all-distilroberta-v1** | **0.789** |
|  |  |

**Based on the above results, model - “E5 Small V2” performs better among all other models that we used.**

**Chunking Strategies on e5-small-v2:**

|  |  |
| --- | --- |
| Method | Score |
| **Batching** | **0.32** |
| **Hybrid Chunking** | **0.34** |

**Based on above strategies used “Hybrid Chunking” gives better results**

**Searching Techniques:**

|  |  |
| --- | --- |
| Method | Score |
| **Default** | **0.41** |
| **Hybrid Search** | **0.47** |

**Based on the above techniques used “Hybrid Search” performs better.**