Enhancing Language Models with Retrieval-Augmented Generation A Comparative Study on Performance

MARKO GRABULOSKI, ALEKSANDAR KARADIMCE, ANIS SEFIDANOSKI
Information Sciences and Technologies
University of Information Science and Technology ”St. Paul the Apostle”
Partizanska bb, 6000 Ohrid
NORTH MACEDONIA

Abstract: Retrieval-Augmented Generation (RAG) is a powerful technique that enhances the capabilities of Large Language Models (LLMs) by integrating information retrieval with text generation. By accessing and incorporating relevant external knowledge, RAG systems address the limitations of traditional LLMs, such as memory constraints and the inability to access up-to-date information. This research explores the implementation and evaluation of RAG systems, focusing on their potential to improve the accuracy and relevance of LLM responses. It investigates the impact of different LLM types (causal, question-answering, conversational) and retrieval-augmentation strategies (sentence-level, paragraph-level) on the performance of RAG systems. We conducted experiments using various open-source LLMs and a custom-built RAG system to assess the effectiveness of different approaches. The findings indicate that RAG systems can significantly enhance the performance of LLMs, especially for complex questions that require access to diverse information sources. T5 conversational models, in particular, demonstrate strong performance in synthesis-based tasks, effectively combining information from multiple retrieved documents. However, causal and question-answering models may struggle with complex reasoning and synthesis, even with RAG augmentation.

Key-Words: Retrieval-Augmented Generation (RAG), Large Language Models (LLMs), Knowledge Retrieval, Contextual Embeddings, Information Retrieval, Neural Networks, LLM Transformer Models, Vector Databases

Received: May 19, 2024. Revised: December 11, 2024. Accepted: January 13, 2025. Published: April 2, 2025.

1 Introduction

Large language models (LLMs) are quite useful and have taken the world of AI by storm. They have extensive usages, and we are still rediscovering what they can be used for. However, there are some aspects that are not ideal, for example:

- **Limitation in Memory**: LLMs are trained once on a set of data. Once trained, they cannot answer questions or generate text on data they have not been trained on (problem of underfitting and high bias).
- **Hardware Requirements and Cost**: Running useful general-purpose LLM requires serious hardware capabilities, which comes with additional costs.
- **Privacy**: Large and useful LLMs are typically run by companies, requiring users to send data to externally managed servers. This may pose problems for individual users who need to submit private data and for companies, especially with GDPR requirements that might not be satisfied.

Extremely important is the first point as it consequently leads to not up-to-date information been stored in the LLMs. This in reality means that the LLMs will not be able to answer up-to-date questions or even worse, will answer questions inaccurately or falsely. This causes a problem, as it introduces limitation on the type of systems that LLMs can be used in.

The paper is motivated by the need to address this LLMs problems. It aims to point out the challenges associated with LLMs and offer solutions using retrieval augmentation generation techniques (RAG). Further on it will explore different RAG techniques and how they can be used to enhance the LLMs. For the purpose of testing the properties and performance of different LLMs in a RAG system in different scenarios, a RAG system is implemented. The entire code, experiment definitions and results can be found in a github repository, [1].

2 What is Retrieval Augmented Generation

Retrieval-augmented generation (RAG) is an advanced artificial intelligence (AI) technique that combines information retrieval with text generation, allowing AI models to retrieve relevant information from a knowledge source and incorporate it into *generated text*, [2].

The basic idea around a RAG system is extending the capabilities of an already trained machine learning model (MLM) in particular an LLM.

One common problem in LLMs is that in order to get a correct output/answer the LLM needs to be trained on a dataset that is relevant to the input/question. For example, asking an LLM who is the winner of 2024 euro championship will not result in a correct answer if the model is trained before 2024. Regardless of the capabilities of the chosen LLM, it just does not contain memory related to this information. In case like this the LLM will need to be retrained. Different problem that occurs in LLMs and machine models in general is that perhaps the LLM is not powerful enough for the given task. Example for the second case is that if a model is asked extremely difficult question, even though the model has been trained on all the relevant data it might still not produce correct output as it does not contain enough memory or parameters to learn the test. This problem in machine learning is known as underfitting.

RAG systems do not suffer as much as other machine learning techniques from this problem, their memory can be extended without the need for re-training or adding additional parameters.

The system has two types of memory:

- **Parametric memory** - refers to the knowledge that is stored a neural network's parameters. In the case of RAG it is a neural network that is part of the LLM model that the RAG system uses.
- **Non-parametric memory** - stored in special types of databases. This non-parametric memory can be extended with additional knowledge easily, and it still can be used by the LLM.

  This is the core idea in RAG. Without any additional training new knowledge can be made available to the system.

The RAG technique integrates a retrieval model and generative model. RAG systems usually work around extending the capabilities of NLP(Natural Language processing models) giving them additional memory or information on various topics. Therefor a RAG system's main two component or sub-systems are:

- **Retrieval and Augmentation component**, manages the storing and retrieval of data from and to the parametric memory. Synthesizes the data and creates a context.
- **Generative AI component**, also called a generator, it is usually a component that contains language generation tool like LLMs. It generates text, answers based on the data provided by the retrieval and augmentation system.

From the perspective of usage of the RAG system, the answer generation is done in three phases:

- **Phase 1 Knowledge retrieval**, when asked a question the RAG retrieval system will retrieve all known information by submitting query to the non-parametric memory.
- **Phase 2 Augmentation**, with the retrieved information, a context around the question is created, the context should contain all information for answering the question.
- **Phase 3 Generation**, the context and question will be passed to an LLM that will generate an answer.

Usually phase one and two are part of the retrieval and augmentation component. The simplified RAG process can be seen in the Figure 1.

```
Question
    |
    ---> | Retrieval and |
         | Augmentation  |
               |
               --> (context + question)
                              |
                              ---> | Generative |
                                   |     AI     |
                                   |  Component | --> Answer
```

Fig. 1: RAG's system retrieval and augmentation phase

3 Generative AI Component

At its core is an LLM transformer. The transformer has two main subcomponents, encoder and decoder. The input text is initially tokenized, meaning it is split into small continuous lists of characters. In the next step, the model transforms the tokens into fixed-size vectors called embeddings. In the final step, in a so-called attention layer, an additional linear transformation on the embeddings is performed, making them dependent on the context they appear in. These context-dependent embeddings, also called contextual embeddings, are the output of the encoder and the input to the decoder. The encoder is of extreme importance in this text as it is a crucial part of a RAG system.

Embeddings are vector representations, typically produced by a neural network, whose main objective is to map (embed) the input media item into a vector space, where the locality encodes the semantics of the input, [3].

The decoder takes the input and generates new embeddings and again combines the output with the embedded tokens to produce new tokens. The vector transformation where token embeddings are transformed to different ones to capture the context of the input text is called attention. It is described in the paper ”Attention Is All You Need”. The result of that research paper, suggests that the LLM transformers do not suffer from problems related to context maintaining as much as recurrent networks. Transformers also outperform the best previously reported models in translation, [4].

The quality of the generated text generally depends on the number of the parameters of the model. Regardless of the quality of the output, every LLM for a given input is expected to generate grammatically correct text and semantically related to the input. For example if I ask even the smallest LLM ”What is the capital of France?” it might return ”Berlin is the capital of France”. The answers will perhaps not be correct, but they will contain the context of the question and grammatically correct.

The generator component of the RAG system needs to be able for a given input and context to generate grammatically correct sentences related to the given context.

This is exactly what is achieved with the LLM transformers and that is why they are ideal to be used in the generation phase of the RAG system.

4 Retrieval and Augmentation Component

The retrieval and augmentation component is used to store and retrieve knowledge outside the knowledge maintained in the LLM. The knowledge here is in the form of paragraphs, sentences, or other forms of continuous text, referred to here as a text block. The text blocks are stored as embeddings.

This part of the system is composed of two components or modules:

- Sentence Embedding Model - AI model
- Non-parametric memory

4.1 Storing Phase

The idea of the storing phase is for a given text block to create embedding and store this embedding in the non-parametric memory. The process is shown in Figure 2.

The process is as follows:

1. Takes block of text, sentence, or paragraph.
2. Passes this text to the sentence embedding model to get an embedding(dense vector).
3. Stores this embedding in a non-parametric memory.

```
Sentence
|
 -> Tokenization
      |
       -> Embedded Vectors
            |
             -> Contextual Vectors
```

Fig. 2: RAG retrieval system, storing phase

4.2 Retrieval Phase

The idea of the retrieval phase is to return all relevant information from the non-parametric memory for a given text, query, or question. The process is shown in Figure 3.

The process is as follows:

1. Takes input text, referred to as a query.
2. Passes the query text to the sentence embedding model to get a query embedding, query dense vector.
3. Using the query embedding, fetches a list of related knowledge from the non-parametric memory.

```
Question
  |
  -> Sentence Embedder
  |          |
  |          -> Embedding
  |              |
  |              -> Non-Parametric memory
  |                     |
  |                     -> Retrieved Documents
  |                             |
  |                             -> Context
  |                                   |
  |                                   -->| Generative |
  ------------------------------------------>| Component  |
                                                 |
                                                 -> Answer
```

Fig. 3: RAG retrieval system, retrieval phase

4.2.1 Non-parametric Memory - Vector Databases

The non-parametric memory is usually a dense vector database. Besides being able to store vectors, a vector database must also have the capability to query for the k the closest vectors to a given input vector. Like in any other traditional database, fast storage and querying are important. Ideally, these databases should be able to store data on disk to enable horizontal scaling and data distribution, which are essential for a RAG system to achieve scalability compared to a traditional LLM model.

The experiment in this research will use the Faiss library, as it satisfies the conditions for vector storage and querying the k closest vectors, it lacks automatic distributive scaling, but a full DB solution is not required for the project as the paper focuses on comparison on LLMs and not on non-parametric memory storage.

The Faiss library is dedicated to vector similarity search, a core functionality of vector databases. Faiss is a toolkit of indexing methods and related primitives used to search, cluster, compress, and transform vectors, [3].

The Faiss database was tested with 768-dimensional ContrieverDb (name of the database from where vectors are imported) dense vector embeddings with up to 1M vectors with 64 bytes per dimension and also with 100M vectors from DeepDb that have a dimension of 96 and 8 bytes per dimension, [3].

Popular vector databases In recent years, there have been significant developments in the field of vector databases. Several new dense vector databases have emerged, and some older databases, like NodeDB and MongoDB, have begun incorporating vector capabilities.

Currently, (as of 2024), these are the options for dense vector storage:

- Redis Vector Similarity Search (VSS)
- Pinecone - exclusively managed, closed source
- Weaviate - open source
- Milvus - open source
- Qdrant - open source
- Vespa - open source
- Cloudflare Vectorize - exclusively managed, closed source
- MongoDB Atlas - fully managed, closed source
- Postges, pgvector - open-sourced

The previous information is taken form various blog posts, [5], [6], [7], [8], [9], [10], [11], [12].

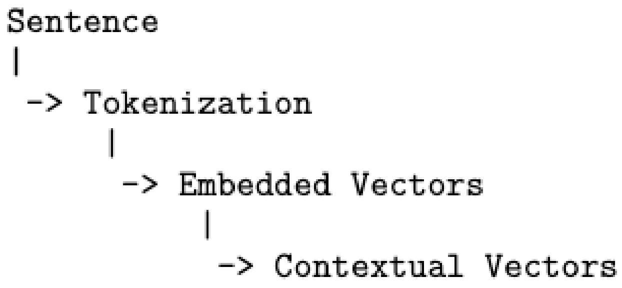
4.2.2 Sentence Embedding Models

Sentence embedding models are derived from LLM transformer models. LLMs transformer contains two general components, encoder and decoder. As described in the section ”Generative AI component”, the input of an encoder is a text, the output is a contextual embedding.

From particular interest for the sentence embedding models and the RAG system is the encoder layer of the LLMs transformer.

In Figure 4 is a description on how sentences are converted to embedding by an encoder.

LLM Transformer's encoder

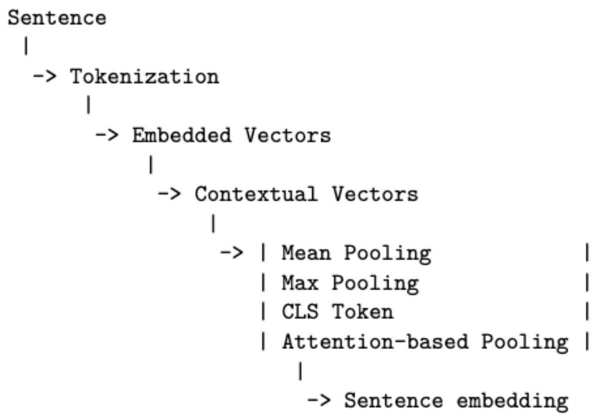


```
Sentence
|
 -> Tokenization
      |
       -> Embedded Vectors
            |
             -> Contextual Vectors
```

Fig. 4: LLM Transformer's encoder

The Sentence embedding models incorporate the LLM transformers encoder layer and adds additional layer of *pooling*. This layer operates on the token embedding and groups them in one embedding. This outputs embedding are dependent on the context of the input text, and are called sentence contextual embedding or sentence embedding. All the sentence contextual embedding have the same size, and it is equal to the size of the token embedding.

Visual description on the process of creating sentence embedding by a Sentence embedding model is shown in Figure 5.



```
Sentence
 |
  -> Tokenization
       |
        -> Embedded Vectors
             |
              -> Contextual Vectors
                   |
                    -> | Mean Pooling            |
                       | Max Pooling             |
                       | CLS Token               |
                       | Attention-based Pooling |
                          |
                           -> Sentence embedding
```

Fig. 5: Sentence embedding model

4.2.3 Pooling

The pooling phase involves the task of combining multiple contextual token embeddings into one sentence embedding. The dimension of the token embeddings is the same as the created sentence embedding. Here are some of the popular pooling methods used in sentence embedding models:

- Mean Pooling
- Max Pooling
- CLS Token
- Attention-based Pooling

The most representative method is Mean Pooling, which calculates an embedding using the mean:

$$meanpool(e_1, e_2, \ldots, e_n) = \frac{1}{n}\sum_{i=1}^{n} e_i \qquad (1)$$

Mean pooling, max pooling, and CLS Token are commonly used techniques, [13].

The goal of sentence embedding models is to produce an embedding or vector from text. As mentioned before, they are built by adding a layer to the LLM's transformer encoder component. The LLMs encoder produces token embeddings, where the sentence embedding models produces embeddings for entire sentences or text blocks.

The requirement of the sentence embedding model is as follows: For every three blocks of text A, P, N, the model SE is to provide three embeddings $SE(A)$, $SE(P)$, $SE(N)$,
such that: if A and P are semantically more similar A and N, then the distance, D:

$$D(SE(A), SE(P)) < D(SE(A), SE(N)) \qquad (2)$$

The simple interpretation is that the sentence embedding model takes a block of text and captures its meaning or semantics by describing it as a vector.

This is an extremely useful feature because it allows for the quantitative representation of text or sentence semantics. In contrast, traditional information theory represents information as the average number of bits transferred, focusing on the quantity of information without considering semantic meaning. LLM embeddings, however, numerically represent the quality of information.

4.2.4 Siamese Training Network and Loss Functions

Up to this point, I have explained how a sentence embedding model works, including its input and output. Even though the input and output are in the required format, sentence embedding models are additionally trained to meet the semantic requirements defined above.

The main issue during training is that the exact output for a block of text is not really known; it can be any embedding. It is the relative distance between the embeddings that is important. Semantically close embeddings should be closer than embeddings that have different meanings. To solve this problem, the model is trained using a Siamese network or a variation of the Siamese network. Initially, the sentence embedding model is trained with a Siamese network that conceptually contains two copies of the same neural network. These two networks share the same parameters. The network ends with a loss function. Popular loss functions are contrastive loss and the triplet loss function.

Contrastive loss function Contrastive loss function uses the Siamese network, shown in Figure 6. The training set elements consist of two text blocks A, C where:

- A - is called an anchor case, it is used to compare against
- C - means positive or negative case, is either a text similar to A or a text block that is not similar to A

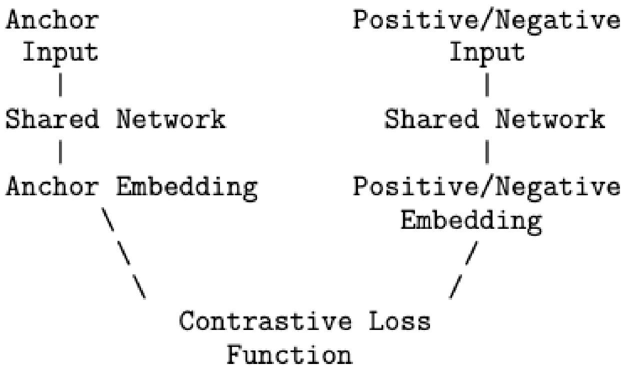


Fig. 6: Siamese network with Contrastive Loss function

The networks end's with a contrastive loss function that is given with the following formula:

$$Loss = \frac{y*D(A,C)^2+(1-y)*max(0,m-D(A,C))^2}{2} \qquad (3)$$

In the previous function y can be 1 or 0. This is because the network can be trained on cases with similar and dissimilar embeddings.

- For similar embeddings $y = 1$, the form of the loss function is:

$$Loss = \frac{1}{2}D(A, C) \tag{4}$$

- Dissimilar embeddings (y=0), the form of the loss function is:

$$Loss = \frac{1}{2}max(0, m - D(A, C))^2 \tag{5}$$

  In this case m acts as a kind of tolerance level and the function is triggered only for a distance bigger than m.

Triplet Loss function Similar to the contrastive loss function training method, the triplet loss function uses a variation of the siamese network where instead of two networks, three networks are used. One single test case consists of three text blocks: A, P, and N. The process is shown in Figure 7.

- The text block A is used for comparison and is called an anchor.
- The text block P is semantically similar to A and is called the positive case.
- The text block N is semantically different from A and P and is called the negative case.

```
Anchor        Positive      Negative
 Input         Input         Input
   |             |             |
Shared         Shared        Shared
Network        Network       Network
   |             |             |
Anchor        Positive       Negative
Embedding     Embedding     Embedding
      \          |           /
       \         |          /
        \        |         /
         \       |        /
         Triplet Distance
           Loss function
```

Fig. 7: Siamese network with triplet loss function

This siamese network ends with a triplet distance activation function. The function calculates the distance $D(A, P)$ $D(A, N)$ and is activated when:

$$D(A, P) > D(A, N) + margin \tag{6}$$

The activation function from the perspective of the sentence embedding model is a Loss function, triplet loss function (Equation 11), and its output is used for training, [13].

$$Loss = max(0, D(A, P) - D(A, N) + margin) \tag{7}$$

Generally the Equation 7 does the following:

- If $D(A, P) < D(A, N)$, the function returns 0 then no adjustments should be made for the current test case.
- If $D(A, P) > D(A, N)$, then the function returns a value greater than 0, this indicates that the model's parameters should be updated, adjusted to minimize this distance.

The $margin$ is set by the user. It introduces a bias to the function. Since the model's parameters are updated when the function returns 0, the $margin$ makes sure that the model is updated more often. To be exact, the model will not be updated only if the distance between the anchor and the positive case is smaller than the distance between the anchor and the negative case plus the $margin$. The $margin$ in a way increases the model's precision. The $margin$ is set by the user and introduces a bias to the function. Since the model's parameters are updated when the function returns 0, the $margin$ ensures that the model is updated more often. Specifically, the model will not be updated only if the distance between the anchor and the positive case is smaller than the distance between the anchor and the negative case plus the $margin$. In this way, the $margin$ helps increase the model's precision.

Contrastive vs Triplet Loss function Both loss functions are used for training the embedder to detect semantic similarities. The primary difference is that contrastive loss is easier to implement in the context of generating the training set, as it requires two text chunks, whereas triplet loss requires three. On the other hand, triplet loss often provides better results during training. Typically, the embedder is trained twice: initially with a contrastive loss function and later fine-tuned with a triplet loss function.

An important point to mention is that the distance function in the triplet or contrastive loss can vary depending on the use case of the network. When trying to find semantic similarities, a cosine distance between the embeddings is used, whereas in different contexts, a norm distance, such as Euclidean distance, can be used.

Cosine distance is based on cosine similarity, or the cosine of the angle between the two embeddings. Cosine distance does not measure the euclidean distance between the embeddings but how similar in direction are they. When the embeddings have similar directions the cosine distance between then is close to 0. Orthogonal embeddings have cosine values equal to 0, and embeddings that are oppositely directed have cosine values close to -1. The loss function needs to be a ”distance-like” measure, so just a cosine similarity between the embeddings won’t be enough since it does not satisfy the distance requirements mainly because it produces negative values. That’s why cosine distance is used:

$$D(A, B) = 1 - \cos(\theta) \tag{8}$$

Where θ is the angle between A and B.

At this point, all the components of the sentence embedding model are explained. A representative example of a sentence embedding model is SBERT, which is based on the popular LLM BERT. This resource will use SBERT as a sentence embedding model.

5 Related Work

This research primarily aims to compare the performance of various LLMs within a RAG system, with a focus on smaller models that can operate on consumer-accessible hardware. A secondary goal is to aggregate and present the advantages of RAG systems, particularly in comparison to standard LLMs or, more broadly, traditional neural network models.

5.1 General RAG Systematization

5.1.1 Naive RAG

This is the simplest form of RAG, it contains all the components that were previously described. This categorization is based on the different components that are part of the RAG system, not their individual complexity. For example naive RAG might contain a generator component with a simple or complex LLM, simple or complex database, or simple or complex embedder. On the other side, this type of RAG will not use complicated server architecture or complex multiphase retrieval or augmentation.

5.1.2 Advanced RAG

Focuses on improving the retrieval of embeddings or data. It adds additional **pre-retrieval** and **post-retrieval** phase.

In the pre-retrieve phase several optimization methods are employed, like:

- Query optimization
- Storage optimization and retrieval
- Extending the stored data with metadata

In the **post-retrieval** phase idea is to create more usable context from the retrieved data. The data now is re-ranked and compressed, cleaned up.

5.1.3 Modular RAG

This type of architecture tries to split the system in to more modules that can be independently scaled and optimized, [14]. Example:

- Query Processing Module
- Retriever
- Re-ranking
- Context Management
- Generation Module

5.2 Different RAG Methods and Models

5.2.1 RAG-Sequence Model

This type of RAG retrieves k embeddings, but unlike the standard RAG implementation, it does not use all the embeddings at the same time to generate the answer. Before returning a token, the next token is generated k times, meaning one token is generated for every embedding. Then, the token with the highest probability is returned. This process is repeated for every token.

The RAG-Sequence model uses the same retrieved document to generate the complete sequence. Technically, it treats the retrieved document as a single latent variable that is marginalized to get the seq2seq probability p(y|x) via a top-K approximation. Concretely, the top K documents are retrieved using the retriever, and the generator produces the output sequence probability for each document, which are then marginalized, [15].

5.2.2 RAG-Token Model

Interesting research has been done using the so-called *RAG-Token Model*, [15]. The RAG technique described, initially retrieves k documents related to the question then augments and creates context for the question and sending the context and question as input to an LLM or the generator. The RAG-Token Model retrieves new documents on every generated token. Once a token is generated, the token is appended to all previously generated tokens which are then appended to the initial question. This results in a string of the following form: ”Question + GenerateTockenList”. This new string is then used again as an input in the RAG, or as an input for the RAG retrieved. The process retrieves new documents for every new generated token and uses these documents to generate the next token.

RAG-Token Model, in the RAG-Token model we can draw a different latent document for each target *token and marginalize accordingly. This allows the generator to choose content from several documents when producing an answer. Specifically, the top K documents are retrieved using the retriever, and then the generator produces a distribution for the next output token for each document, before marginalizing, and repeating the process with the following output token*, [15].

5.3 Iterative RAG

One example is the ”Speculative RAG”, [16]. Based on the question it retrieves N embeddings. The embeddings are then clustered in k classes using $K - means$. From the k clusters one embedding is selected per cluster, resulting in k embeddings. Then the algorithm uses two LLMs called the drafter and the verifier. The drafter is used to generate the answer, called a draft and explanation called rationale. It is important to note that not all LLMs generate rationales. There are some that can do this, but usually the model needs to be trained to provide the rationale. The RAG-Model from Hugginface provides explanations that can be used as a rationale, also the retrieved embeddings might be used as rationales. Once the answer and the rational are generated, the quality of the answer and question is measured by the verifier.

5.3.1 Self-Consistency Score

It gives the probability that the LLM can generate the answer and the rationale based on the question. It is given by the joint probability formula:

$$P_{sc} = P(A, B \mid Q) = P(A \mid Q) \cdot P(B \mid Q, A) \quad (9)$$

Where:

- A is the answer
- B is the rationale
- Q is the question

The simple interpretation of Equation 9 is that it calculates the conditional joint probability of the answer being generated based on the question and the probability that the rationale is generated based on the question and the answer. These probabilities are calculated by the LLM.

5.3.2 Self-reflection Score

It tries to determine if the rational supports the answer (Equation 10). For this a new question R can be formed: Does the rational A sports the answer B for the question Q? The new question is called a self-reflected statement and denoted with R. Then the probability of Yes been generated from the LLM is measured. In general, the self-reflection score is the conditional probability of Yes been generated by the LLM given the input of a question Q, answer A, rational B and self reflected statement R.

Self-reflection score:

$$P_{sr} = P(”Yes”|Q, A, B, R) \quad (10)$$

The process is repeated for all the clusters, and a $p_{sr} * p_{sc}$ is calculated. The answer with the highest $p_{sr} * p_{sc}$ score is picked as the most relevant, and it is the answer that is returned.

5.4 Recursive RAG

As any recursive algorithm it repeats its self until the results satisfy certain criteria. In case of the retrieval, a naive RAG system performs good when all the information required to answer the question is in the retrieved embedding, but it fails when this information is scattered in different embeddings. The Recursive RAG starts by retrieving semantically similar documents to the question and continues to search for other similar embeddings to the already retrieved embeddings. In this way, it tries to capture all relevant information, [12].

6 Research Methodology

In this research, a simple RAG system is developed. The system is flexible and modular enough so can use different LLMS as generators. The system will have a retriever that will be able to create embeddings based on sentences and also on paragraphs. The retriever will also retrieve embeddings based on a question/query. All the chosen LLMs and their corresponding RAGs will be asked three different types of Open-domain questions. These types of questions cannot be answered with a simple yes or no and require accessing a broad range of information to provide an answer. Find the RAG implementation and refer to [1].

The purpose of the experiments:

- Compare the quality of answers generated by a RAG system versus a standard LLM for open-domain questions.
- Assess the influence of different LLM architectures (causal, question answering, and T5) on the RAG system’s performance.
- Analyze the effect of model size and retrieval strategies on answer quality in the RAG system.
- Examine how retrieval context size affect the accuracy and relevance of answers.
- Evaluate the efficiency of the RAG system across diverse configurations, including question types and retrieval approaches.

6.1 LLMs

Depending on the type of LLM, three different models with distinct architectures will be used:

1. Causal models process input text sequentially, left-to-right, and predict the next token in the sequence.
2. Question answering models are trained to understand both the context and the question, extracting the answer directly from the provided context.
3. T5 models use a text-to-text framework, a more modern and flexible approach that extends beyond question answering to handle a variety of NLP tasks.

Regarding model size, the experiments will be conducted on a small set of LLMs ranging from 66M up to 1.3B. Model sizes and type are shown in Table 1 (Appendix).

For comparison, the currently used production ChatGPT 3.5, according to ChatGPT, has a size of around 175B. ChatGPT 4.0 has not exposed any information about the number of parameters or architecture. Most probably, both models employ the RAG technique to improve their performance.

It is important to point out that the causal models are not trained for answering questions but for generating text based on input. Running a RAG around causal LLMs will give an idea of how powerful a RAG system can be. The question-answering models are more suitable for use in a RAG system. They extract sequence of text within the context that has the highest probability of being an answer given a specific question. The T5 models are conversational models, and although they are not specifically trained to answer questions, they are trained to maintain a conversation and could potentially outperform the causal and the question-answering LLMs.

6.2 Questions

The questions asked and their types are:

1. Fact-based question - ”How tall is the Pyramid of Giza?”
2. List-based question - ”What materials were used in constructing the Great Wall of China?”
3. Synthesis-based Fact question - ”Which famous structures were either designed or structurally influenced by Gustave Eiffel?”

The ’Synthesis-based Fact’ question is a type of question where the answer needs to be derived from more than one retrieved fact. Unlike the other questions, where the answer is directly contained in the embedding, correctly answering the synthesis-based fact question implies that the retrieval system correctly mapped and retrieved the embeddings, and that the LLM is capable of drawing conclusions based on several facts.

6.3 Retrieval and Augmentation

There will be 3 types of RAG augmentation, in this text called retrieval:

- Retrieval of one sentence, will create a context for the question from one sentence using a facts sentence based database.
- Retrieval of one paragraph, will create a context for the question from one paragraph using a facts sentence based database.
- Retrieval of three paragraphs, will create a context for the question from three paragraphs using a facts-paragraph-based-database. The database for the sentences and paragraphs is created from two files containing semantically correct but not necessarily related sentences and paragraphs.

6.4 RAG Solution

The RAG system is a manual solution, not an existing RAG system. The idea is to explore how even a simple RAG system can extend the capabilities of the LLM.

6.4.1 Sentence Embedding Model

For sentence embedding model a SBART based model will be used.

6.4.2 Non-Parametric Library

For non-parametric library *faiss* will be used, it is more than enough to handle this experiment.

6.4.3 LLM Placeholder

The LLM is part of the generator component, it allows attachment of different LLMs to the RAG solution. The reason for this is to test how different LLMs perform.

6.5 In Memory Database

Key value database will be used to store records, that are hold the embedding as a key and its corresponding text as a value. This will implement as simple in memory database.

6.5.1 Environment

Jupyter notebook.

6.5.2 Hardware

For the particular reasons an open sourced LLMs will be used that can be run locally on the pc. The chosen hardware is a m3 processor with 36GB of RAM.

6.6 System Architecture

The architecture of the RAG developed solution for testing the LLMs can be seen in Figure 8 (Appendix). Where the important processes are numerated and described in the following list:

1. Storing phase, the documents containing facts are passed to the storing phase component with the purpose of permanent storage.
2. Storing phase, storing phase component embedder creates embeddings for the documents and stores them in the non-parametric memory.
3. Storing phase, each vector embeddings with its corresponding text is stored a key-value pair in the in-memory database.
4. Retrieval augmentation phase, the actor, user asks a question, this question is passed to the embedder.
5. Retrieval augmentation phase, the embedder creates an embedding for the question and queries the non-parametric memory for the k-closest embeddings.
6. Retrieval augmentation phase, the retrieved k-closest embeddings are used to obtain their corresponding texts from the in-memory key-value database, forming the context.
7. Generation augmentation phase, the context with the question is passed to the generator component.
8. Generation phase, the generator component inside it creates an answer for the users question.

6.6.1 Experiments Definition

In total five experiments will be performed. The question type, the retrieved type of information(text), and the number of retrievals for each experiment are shown in Table 2 (Appendix).

7 Results and Discussion

7.1 Experiment 1: Fact-based Question - “How Tall is the Pyramid of Giza?”

7.1.1 Experiment Description

In this experiment, various language models were tested to answer the fact-based question: ”How tall is the Pyramid of Giza?”. The experiment parameters are as follows:

- **Question Type**: Fact-based
- **Question**: ”How tall is the Pyramid of Giza?”
- **Retrieval Type**: Sentence retrieval
- **Number of Retrievals**: 1

7.1.2 Causal Models

The results of the experiment are shown in Table 4 (Appendix). In this experiment setup, all the causal LLMs generated incorrect answers. Except for `facebook/blenderbot-90M`, they all generated grammatically correct sentences. Regarding the RAG answers, all models generated correct RAG answers except for `facebook/blenderbot-90M`. This is an amazing result, as without any additional training or tuning, the models using the RAG approach were able to generate answers that contained the correct information. However, the answers lacked structure and provided additional information that was not required. This is due to the nature of the causal models. Taking into consideration the parameters, the smallest model that provided a correct answer was GPT-2 with 90M.

7.1.3 QA Models

The results of the question answering models for this experiment are shown in Table 5 (Appendix). The QA models also provided correct answers. Compared to the causal LLMs RAG generated answer, here the models perform better as a model of 66M parameters was able to answer the question correctly. The answers generated by the LLM were incorrect for every model.

7.1.4 T5 Models

The results of the T5 conversational models this experiment can be seen in Table 6 (Appendix). The T5 models RAG generated answer is correct for the T5-base model and partially correct for the T5-large model. The T5-large model answered correctly in feet but not in meters.

The results of the T5 conversational models from this experiment can be seen in Table 6 (Appendix). The T5 model’s RAG-generated answer is correct for the T5-base model and partially correct for the T5-large model. The T5-large model answered gave the correct answer in feet but not in meters.

Summary: Given the size of the models and the accuracy of the answers, the QA models are most appropriate for answering fact-based Question. They are efficient and are able to provide the most probable sequence inside the context that answers the question.

7.2 Experiment 2: List-based Question - ’What Materials were used in Constructing the Great Wall of China?’

7.2.1 Experiment Description

In this experiment, various language models were tested to answer the list-based question: ’What materials were used in constructing the Great Wall of China?’. The experiment parameters are as follows:

- **Question Type**: List-based Questions
- **Question**: ”What materials were used in constructing the Great Wall of China?”
- **Retrieval Type**: Sentence retrieval
- **Number of Retrievals**: 1

7.2.2 Causal Models

The results of the causal models for the experiment in the current section are documented in Table 7 (Appendix). In all cases, the causal models generated incorrect or irrelevant LLM answers. The RAG approach, however, performed significantly better, with all models providing correct information about the materials used in constructing the Great Wall of China. Because of the causal LLM model’s nature, the RAG approach’s answer contains additional unnecessary information.

7.2.3 QA Models

The results of the QA models for this experiment are documented in Table 8 (Appendix).

The QA models performed well in generating correct RAG answers, accurately listing the materials used in constructing the Great Wall of China. Similar to the first experiment, the generated LLM answers were incorrect, but the RAG system compensated by providing correct and relevant answers, demonstrating its strength in retrieving and presenting accurate information.

7.2.4 T5 Models

The results of the T5 conversational models for this experiment can be seen in Table 9 (Appendix).

Here, the T5 models also demonstrated strong performance in the RAG approach. They correctly listed the materials used in constructing the Great Wall of China. However, like the other models, their direct LLM answers were inaccurate or non-informative, further highlighting the importance of RAG in ensuring accurate output.

Summary:

Given the size of the models the QA models are most appropriate for answering List-based Question. Same as in the previous experiment, the QA models are efficient and provide the most probable character sequence inside the context that answers the question.

7.3 Experiment 3: Fact-based Question - ’How Tall is the Pyramid of Giza?’

7.3.1 Experiment Description

In this experiment, various language models were tested to answer the fact-based question: ’How tall is the Pyramid of Giza?’ using paragraph retrieval with one retrieved document. Storing paragraphs is more efficient than storing sentences, as the size of the database will be smaller, and paragraphs hold more context and information than sentences.

The experiment parameters are as follows:

- **Question Type**: Fact-based Question
- **Question**: ”How tall is the Pyramid of Giza?”
- **Retrieval Type**: Paragraph retrieval
- **Number of Retrievals**: 1

7.3.2 Causal Models

This experiment’s results for the causal models can be observed in Table 10 (Appendix).

In this experiment, the causal models generally provided incorrect or nonsensical LLM answers. However, the RAG approach performed significantly better, producing correct answers by accurately retrieving and presenting the relevant information from the paragraph.

7.3.3 QA Models

The experiment results for the QA models are shown in Table 11 (Appendix).

The QA models continued to perform well, with the RAG approach providing correct and relevant answers. Despite the direct LLM answers being incorrect, the RAG system ensured accurate information was retrieved and presented.

7.3.4 T5 Models

The results of the T5 conversational models for this experiment are shown in Table 12 (Appendix).

The T5 models also demonstrated strong performance in the RAG approach, consistently providing the correct height of the Pyramid of Giza. As seen in previous experiments, the direct LLM answers were inaccurate.

Summary:

The QA models and T5 models were particularly efficient, consistently providing the correct height of the Pyramid of Giza. Given the size of the models, the most efficient for answering fact-based Question with one-paragraph retrieval are the QA models.

7.4 Experiment 4: List-based Question - 'What Materials were used in Constructing the Great Wall of China?'

7.4.1 Experiment Description

In this experiment, various language models were tested to answer the list-based question: 'What materials were used in constructing the Great Wall of China?' using paragraph retrieval with one retrieved document. The experiment parameters are as follows:

- **Question Type**: List-based Question
- **Question**: "What materials were used in constructing the Great Wall of China?"
- **Retrieval Type**: Paragraph retrieval
- **Number of Retrievals**: 1

7.4.2 Causal Models

The results of the causal models for this experiment can be seen in Table 13 (Appendix).

In this experiment, the causal models generally provided incorrect or nonsensical LLM answers. The RAG approach, however, showed improvement, with more accurate information being retrieved and presented. Despite this, the answers still lacked consistency in listing all the materials used in the construction of the Great Wall of China. These models would require additional fine-tuning if they are to be used for question answering tasks.

7.4.3 QA Models

The results of the QA models for the experiment can be seen in Table 14 (Appendix).

The QA models performed exceptionally well in generating correct RAG answers, accurately listing the materials used in constructing the Great Wall of China. The direct LLM answers were not particularly useful, but the RAG system effectively retrieved and presented the necessary information.

7.4.4 T5 Models

The results of the T5 conversational models for this experiment can be seen in Table 15 (Appendix).

The T5 models also performed well in the RAG approach, consistently providing the correct list of materials used in constructing the Great Wall of China. Similar to other models, the direct LLM answers were inaccurate or non-informative, but the RAG system ensured accurate output.

Summary:

The QA models proved to be the most efficient for answering list-based questions with paragraph retrievals. Their performance in generating accurate RAG answers highlights their suitability for RAG-based solutions, especially when detailed and precise information is required. Given their size and efficiency, the QA models are the best choice for implementing a RAG solution in this context.

7.5 Experiment 5: Synthesis-based Question - "Which Famous Structures, Both Designed or Structurally Influenced by Gustave Eiffel?"

7.5.1 Experiment Description

In this experiment, various language models were tested to answer the synthesis-based question: "Which famous structures, both designed or structurally influenced by Gustave Eiffel?" using paragraph retrieval with three retrieved documents. The experiment parameters are as follows:

- **Question Type**: Synthesis-based Question
- **Question**: "Which famous structures, both designed or structurally influenced by Gustave Eiffel?"
- **Retrieval Type**: Paragraph retrieval
- **Number of Retrievals**: 3

The synthesis-based question requires that several facts are taken into consideration in order to answer the question. In this case the retriever will need to provide to the generator several paragraphs with information on Gustave Eiffel. In the answer generation, augmentation phase the RAG needs to combine the facts in to a conclusion and generate the answer.

7.5.2 Causal Models

The results of the causal models for this experiment are stored in Table 16 (Appendix).

In this experiment, the causal models struggled to produce accurate or relevant LLM answers. The RAG approach helped improve the accuracy but was still limited, with most models failing to provide a comprehensive list of structures designed or influenced by Gustave Eiffel. The answers often lacked synthesis, indicating that causal models might not be the best choice for complex synthesis-based questions.

7.5.3 QA Models

The results of the QA models for this experiment are documented in Table 17 (Appendix).

The QA models had mixed performance in this experiment. While the direct LLM answers were often uninformative, the RAG answers managed to identify at least one structure associated with Gustave Eiffel, but they lacked completeness. The synthesis required for this question appeared challenging for these models. Main reason for this behavior is that this QA models are trained to extract one sequential list of characters from the context, that has the biggest probability to answer the question.

7.5.4 T5 Models

The results of the T5 conversational models for the experiment in the current section are documented in Table 18 (Appendix).

The T5 models performed relatively well in the RAG approach, with the larger model (T5-large) correctly listing both the Eiffel Tower and the Statue of Liberty as structures associated with Gustave Eiffel. This highlights the potential of T5-models in synthesis-based tasks, especially when more complex reasoning is required.

Summary:

For synthesis-based questions, the T5 models, particularly the larger variant, demonstrated the most potential in the RAG approach, successfully identifying multiple structures associated with Gustave Eiffel. Although the QA models could partially answer the question, they struggled with the complexity of synthesizing information from multiple documents. The causal models, even with RAG, were not able to handle this task effectively. Therefore, T5 models are the best choice for RAG-based solutions when dealing with synthesis-based questions.

7.6 Summary on Performance

In terms of answer quality, the answers can be categorized into four groups, incorrect, partial, over information correct. A partial answer, would be an answer that does provide some, but not all, of the necessary information to answer the question also answers that are truncated or sentences not fully articulated will fall in this category. Over-information is a correct answer that contains additional or repeatable information that is not required by the question. The answer quality alongside the model's speed and size in parameters can be observed in the tables: Table 19 (Appendix), Table 20 (Appendix), Table 21 (Appendix), Table 22 (Appendix), Table 23 (Appendix) for each experiment. In terms of answer quality overall in all categories the T5-models performed the best. In terms of measurable performance several interesting conclusions can be derived from the data. For the purpose of comparing performance, a simple efficiency score can be defined as:

$$EfficiencyScore = \frac{1}{Speed_n * Parameters_n} \quad (11)$$

Where $Speed_n$ is the normalized speed of the model and $Parameters_n$ is the normalized number of parameters of the model.

The normalized speed and normalized parameters are calculated based on the maximum speed and the maximum number of parameters across all experiments. Higher numbers in Equation 11 indicate better performance. Based on this formula, average efficiency scores can be calculated for each model across all experiments. The results can be seen in Table 3 (Appendix).

Based on this assessment, the QA models are in the top segment of Table 3 in Appendix, their efficiency score ranges from 200,85 to 5550,67, where the 'distilbert-base-uncased-distilled-squad' being the model with the highest efficiency score. Their average answer time is quite fast, starting from 0.17 up to 0.92 seconds per model.

The GPT model family is positioned at the lower end of the table. The model with the lowest average efficiency score of 1.56 is 'EleutherAI/gpt-neo-1.3B'. The average answer is slower than of the others model families, with an average range per model from 3.8 up to 25 seconds. The 'EleutherAI/gpt-neo-1.3B' is the slowest model in terms of absolute answer speed, its average answer time is 25.20 seconds.

The T-model family is positioned in middle of the efficiency score Table 3 in Appendix. They are slower than the QA models. 'T5-base' model has an average efficiency score of 22.13 for the T5-large and 185.50 for the T5-base model. When considering the answer quality, T5-large model performed better than the T5-base model. This is visible in the last experiment where the T5-base model failed to answer correctly on the synthesis based question. This might indicate that model can't handle complex context. Given the size of the model, 22M parameters, this is probably an underfitting problem and using same model type with more parameters might solve this issue. In terms of the absolute time of response, the T5 models are quite fast with an average from 1.2 to 3.5 seconds, this is an acceptable time. Without additional fine-tuning, this model family can be used in a RAG system.

For comparison the expected response time of OpenAI's GPT-4o model is 1-3 seconds for a medium to short answers.

When comparing the LLM answer time to the RAG answer time, two tasks need to be measured. The time to retrieve the context and the time to generate the answer. The time to retrieve the context depends on the parametric memory index, parametric memory size, and number of retrievals. The time to generate the answer depends on the model and the size of input, including the question and context. Larger models need more time to generate the answer. Larger questions and contexts also require more time to be answered. The average response time of the LLM models was 5.6 seconds across all experiments. The average response time of the RAG for the experiments is:

- Experiment 1: 4.2 seconds
- Experiment 2: 4.2 seconds
- Experiment 3: 6.2 seconds
- Experiment 4: 6.0 seconds
- Experiment 5: 9.1 seconds

The context in experiments, starting from experiment 1 to experiment 5 was increased. Experiments 1 and 2 have a context of one sentence, experiment 3 and 4 have a context of one paragraph and experiment 5 has a context of three paragraphs. This data indicates that as stated before response time in a RAG system depends on the context size.

8 Conclusion

This paper has explored the potential of Retrieval-Augmented Generation (RAG) techniques to enhance the capabilities of Large Language Models (LLMs). By combining the strengths of information retrieval and generative models, RAG systems can overcome the limitations of traditional LLMs, such as memory constraints and the inability to access and process real-world information. By integrating information retrieval and generation, RAG systems can significantly improve LLM-generated responses' accuracy, relevance, and factual correctness. This research explores the implementation and evaluation of RAG systems, focusing on the impact of different LLM types (causal, question-answering, conversational) and retrieval strategies (sentence-level, paragraph-level) on their performance. The experiments were conducted using various open-source LLMs and a custom-built RAG system to assess the effectiveness of different approaches. A notable observation in this paper when evaluating the capabilities of LLMs versus RAG systems is that RAG systems offer several advantages:

- **Horizontally Scalable**: RAG systems can efficiently scale by distributing the retrieval process across multiple nodes, allowing for handling larger datasets and more complex queries.
- **Distributive**: RAG systems' modular nature enables the distribution of tasks across different components, enhancing their robustness and flexibility.
- **Mitigates High Bias**: By integrating external knowledge retrieval, RAG systems reduce the need for extremely large models with numerous parameters, thus, to a degree, avoiding issues related to underfitting and high bias in terms of the real-world dataset or the cross-validation set where the model is not flexible enough or lacks sufficient curvature to approximate the problem.
- **Performance Enhancement**: By leveraging external knowledge, RAG systems can surpass the performance of their base generative models, resulting in more accurate and contextually relevant responses.

For example, a decent quality question-answering system can be created using only a 66M parameter pertained question-answering model like 'DistilBERT' when using an index like 'FAISS'. However, the paragraphs and sentences provided to the model must contain the answer explicitly written, with no pronouns or ambiguous references. In the case of the T5 models, they perform better than QA in more complex text scenarios; these models can understand and synthesise information.

The 't5-large_770M' given its size and performance would be a good candidate for a more general RAG solution. When choosing an LLM, it's important to pick a large enough model. This ensures it can handle complex tasks and will not suffer underfitting. A good strategy when choosing between models from the same model family is to pick the model with the most parameters and an acceptable average response time. Models with more parameters will usually perform better in terms of answer quality. An efficiency score for comparison could also be useful in the decision-making process. The T-5 models are solid candidates for a self-hosted production RAG system. Designing a horizontally scalable system with multiple T-5 model nodes and a distributed parametric memory index would be a feasible solution. The findings indicate that RAG systems can significantly enhance the performance of LLMs, particularly in tasks that require access to external knowledge. T5 conversational models, in particular, demonstrate strong performance in synthesis-based tasks, where they can effectively combine information from multiple retrieved documents. However, causal and question-answering models may struggle with complex reasoning and synthesis, even with RAG augmentation. Popular cloud-based, pre-existing LLMs like OpenAI's GPT-4 or GPT-3 models can also be used when developing RAG systems in environments where data privacy is not a concern. This type of RAG system will easily outperform any open-source LLM-based RAG solution. More and more companies are offering their powerful LLMs as APIs on a subscription or pay-per-use basis.

As a practical software solution for real-world applications, a RAG system can be implemented to power question-answering systems where knowledge is stored as documents, text, or any other storage system—not necessarily a traditional database. These systems are particularly suitable for chatbots, such as those used for customer support across various industries. In medicine, they can assist doctors by providing precise dosing guidelines for medications or warning about potential side effects of combining drugs. In IT, RAG systems can function as knowledge management tools, storing comprehensive documentation about how systems operate and enabling engineers to query and retrieve insights. A key advantage of these systems is their ability to be easily updated with new knowledge without the need to retrain the underlying model.

In conclusion, the main challenge in designing a RAG (Retrieval-Augmented Generation) system is not the LLM (Large Language Model) itself but the context creation process. The model can only respond accurately if the context contains all the information required to answer the question. On the other hand, as shown in this paper, if the context is too large, the model's response time will be slower. Research and advancements in information retrieval aimed at creating the minimum required context for a question are crucial for developing RAG systems and the broader field of AI.

In summary, RAG systems provide a scalable, distributive, and cost-efficient solution to enhance the capabilities of LLMs. They address key limitations and improve overall performance without needing excessively large and complex models.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used Grammarly and ChatGPT to correct grammatical and syntactical errors, which improved the correctness and readability of the study. After using these tools/services, the author(s) reviewed and edited the content as needed and take full responsibility for the content of the publication.

References:

[1] M. Grabuloski, (2024). ”RagSystems”, *GitHub repository* RagSystems, 2024.

[2] P. Miesle, ”Retrieval-Augmented Generation (RAG) Explained: Understanding Key Concepts,” *DataStax*. Retrieved from DataStax Guide on RAG.

[3] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, ”The Faiss Library,” *arXiv preprint arXiv:2401.08281*, 2024. pp. 1-13.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, ”Attention Is All You Need,” *arXiv preprint arXiv:1706.03762*, 2017, pp. 8-10.

[5] M. Ali, ”Optimizing RAG: A Guide to Choosing the Right Vector Database,” *Medium*, 2023. Retrieved from Medium Guide on RAG Databases.

[6] Cloudflare Docs, ”Best Practices: Insert Vectors,” *Cloudflare*. Retrieved from Cloudflare Guide on Vector Insertion.

[7] K. Kehrer, ”Best Vector DBs for Retrieval Augmented Generation (RAG),” *Aporia*, 2023. Retrieved from Aporia Guide on Vector Databases for RAG.

[8] Databricks, ”Retrieval-Augmented Generation (RAG),” *Databricks*. Retrieved from Databricks Glossary on RAG.

[9] MongoDB Docs, ”Atlas CLI: Deploy Local,” *MongoDB*. Retrieved from MongoDB Atlas CLI Documentation.

[10] V. M. W. Benedict, ”RAG with Atlas Vector Search and MongoDB,” *Medium*, 2023. Retrieved from Medium Guide on RAG with Atlas.

[11] T. Bratanič, ”Neo4j Langchain Vector Index Implementation,” *Neo4j*, 2023. Retrieved from Neo4j Blog on Langchain Vector Index.

[12] Pondhouse Data, ”Advanced RAG: Recursive Retrieval with LlamaIndex,” *Pondhouse Data*, 2023. Retrieved from Pondhouse Guide on Advanced RAG.

[13] N. Reimers and I. Gurevych, ”Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” *arXiv preprint arXiv:1908.10084*, 2019. pp. 1-11

[14] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, and H. Wang, ”Retrieval-Augmented Generation for Large Language Models: A Survey,” *arXiv preprint arXiv:2312.10997*, 2023. pp. 1-13

[15] P. Lewis, W. Yih, T. Rocktäschel, and S. Riedel, ”Retrieval-Augmented Generation (RAG) Explained: Understanding Key Concepts,” *arXiv preprint arXiv:2005.11401v4*, 2020. pp. 3

[16] Z. Wang, Z. Wang, L. Le, H. S. Zheng, S. Mishra, V. Perot, Y. Zhang, A. Mattapalli, A. Taly, J. Shang, C.-Y. Lee, and T. Pfister, ”Speculative RAG: Enhancing Retrieval-Augmented Generation through Drafting” *arXiv preprint arXiv:2407.08223*, 2024.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

Marko Grabuloski has provided the conceptualization and completed the data load, cleaning, modeling and analysis in this research study.
Aleksandar Karadimce and Anis Sefidanoski provided mentoring, supervision, and guidance for the complete manuscript, as well as final review and editing.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received to conduct this study.

Conflicts of Interest

The authors have no conflicts of interest to declare that they are relevant to the content of this article.

Appendix

Table 1. Model Descriptions

Model	**Description**	**Number of Parameters**	**Type**
Blenderbot 90M	Facebook AI's BlenderBot Model	90M	Causal Model
GPT-2 124M	OpenAI's Generative Model	124M	Causal Model
GPT-Neo 125M	EleutherAI's Generative Model	125M	Causal Model
GPT-2 Medium 355M	OpenAI's Generative Model	355M	Causal Model
GPT-2 Large 762M	OpenAI's Generative Model	762M	Causal Model
GPT-Neo 1.3B	EleutherAI's Generative Model	1.3B	Causal Model
DistilBERT	Hugging Face's Optimized BERT for QA	66M	Question Answering Model
roberta-base	Facebook AI's Optimized BERT for QA	125M	Question Answering Model
bert-large-uncased-whole-word-masking-finetuned-squad 340M	Google's NLU Model for QA	340M	Question Answering Model
T5 Base 220M	Google's Text-to-Text Transfer Transformer	220M	T5
T5 Large 770M	Google's Text-to-Text Transfer Transformer	770M	T5

Table 2. Experiment description

Experiment	**Question Type**	**Augmentation Type**	**Augmentations**
1	Fact-based question	Sentence	1
2	List-based question	Sentence	1
3	Fact-based question	Paragraph	1
4	List-based question	Paragraph	1
5	Synthesis-based Fact question	Paragraph	3

Table 3. Model's Average Response Time and Efficiency Score

Model Name	**Avg. Response Time**	**Avg. Efficiency Score**
distilbert-base-uncased-distilled-squad	0.1672	5550.6793
deepset/roberta-base-squad2	0.3384	1511.1777
facebook/ blenderbot-90M	2.2683	252.6381
bert-large-uncased-whole-word-masking-finetuned-squad	0.9152	200.8508
t5-base	1.2539	185.5049
gpt2	3.8098	109.4958
EleutherAI/ gpt-neo-125M	4.0386	103.6577
t5-large	3.5096	22.1266
gpt2-medium	8.9408	16.7129
gpt2-large	15.0616	4.6237
EleutherAI/gpt-neo-1.3B	25.2002	1.5673

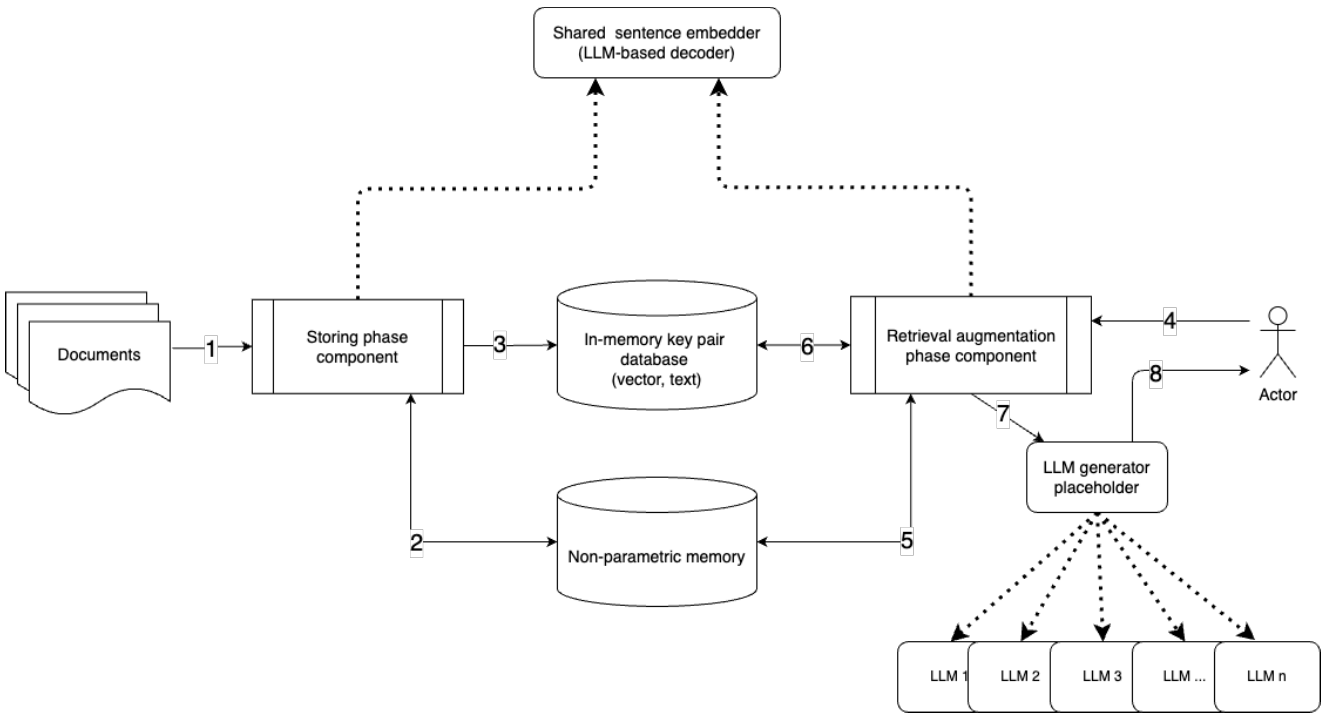

Shared sentence embedder (LLM-based decoder)
Documents
1
Storing phase component
3
In-memory key pair database (vector, text)
6
Retrieval augmentation phase component
4
Actor
8
7
LLM generator placeholder
2
Non-parametric memory
5
LLM 1
LLM 2
LLM 3
LLM ...
LLM n

Fig. 8: RAG system diagram

Table 4. Fact-based question, single sentence retrieval, causal models

LLM	**LLM Answer**	**RAG Answer**
facebook blenderbot-90M	balkicker mutations mutations pment mutations mutations mutations kicker fiers eston eston eston eres bender eston …	lease willis lease mative mative lease willis mative lease lease lease willis …
gpt2	The Pyramid of Giza is the tallest building in Egypt. The Pyramid of Giza is located in the Giza Plateau …	The Great Pyramid of Giza is the largest and oldest of the three pyramids, standing at 146.6 meters …
EleutherAI gpt-neo-125M	The Pyramid of Giza is the tallest pyramid in the world. It was completed in the 13th century AD …	The Pyramid of Giza is the largest and oldest of the three pyramids, standing at 146.6 meters …
gpt2-medium	The Pyramid of Giza is the tallest structure in Egypt. It is 1,068 feet tall, and is one of the largest pyramids …	The Pyramid of Giza is the largest and oldest of the three pyramids, standing at 146.6 meters …
gpt2-large	The Pyramid of Giza is the tallest structure in the world. How tall is the Taj Mahal? …	The Great Pyramid of Giza is the largest and oldest of the three pyramids, standing at 146.6 meters …
EleutherAI gpt-neo-1.3B	The Pyramid of Giza is one of the Seven Wonders of the Ancient World. It is one of the Seven Wonders of the Modern World …	The pyramid of Giza, also known as the Pyramid of Khufu … and oldest of the three pyramids, standing at 146.6 meters …

Table 5. Fact-based question, single sentence retrieval, question answering models. ”How tall is the Pyramid of Giza?”

LLM	**LLM Answer**	**RAG Answer**
distilbert-base-uncased-distilled-squad_66M	how tall is the pyramid of giza? [SEP]	146.6 meters
deepset roberta-base-squad2_125M		146.6 meters (481 feet)
bert-large-uncased-whole-word-masking-finetuned-squad_340M	tall	146.6 meters

Table 6. Fact-based question, single sentence retrieval, T5 conversational models. ”How tall is the Pyramid of Giza?”

LLM	**LLM Answer**	**RAG Answer**
t5-base_220M	None	146.6 meters (481 feet)
t5-large_770M	False	140 meters (481 feet)

Table 7. List-based question, single sentence retrieval, causal models. ”What materials were used in constructing the Great Wall of China?”

LLM	LLM Answer	RAG Answer
facebook blenderbot-90M	pos sitter sitter drip ba ba ba le diving ba ba bee ba ba da ba ba cats ba ba table da ba da disc ba ba orba da da prba ba prda da **end**	scibal pubalfatty composition asses sciscifatty fatty bee pumubee bee mubee zarbalpuscibal mutbal bee bal composition fatty composition fatty bee bal strip scimutcomposition **end**
gpt2	The Great Wall of China is the largest in the world, and is the largest city in the world. It is the largest city in the world. It is the largest city in	The Great Wall of China is made of various materials, including stone, brick, tamped earth, and wood. The Great Wall of China was built by the People’s Republic of …
EleutherAI gpt-neo-125M	The Great Wall of China was constructed by the Chinese government in the 17th century, and is known as the Great Wall of China …	The Great Wall of China is made of various materials, including stone, brick, tamped earth, and wood. Answer What materials were used in constructing the Great Wall of China?
gpt2-medium	The Great Wall of China was built by the Ming Dynasty (1644-1911). The Great Wall was constructed of a combination of stone, wood, and metal …	’Answer The Great Wall of China was built of various materials, including stone, brick, tamped earth, and wood. The Great Wall of China was constructed of various materials, including stone …
gpt2-large	The Great Wall of China was built by the Qin Dynasty (221-206 B.C.), which lasted from 221 to 206 B.C. …	’The Great Wall of China is made of various materials, including stone, brick, tamped earth, and wood.’Answer What materials were used in constructing the Great Wall of China?
EleutherAI gpt-neo-1.3B	The Great Wall of China was constructed using many different types of materials, including wood, stone, brick, clay, and iron …	The Great Wall of China is made of various materials, including stone, brick, tamped earth, and wood.

Table 8. List-based question, single sentence retrieval, question answering models. ”What materials were used in constructing the Great Wall of China?”

LLM	LLM Answer	RAG Answer
distilbert-base-uncased-distilled-squad_66M	what materials were used in constructing the great wall of china? [SEP]	stone, brick, tamped earth, and wood
deepset roberta-base-squad2_125M		stone, brick, tamped earth, and wood
bert-large-uncased-whole-word-masking-finetuned-squad_340M		stone, brick, tamped earth, and wood

Table 9. List based question, single sentence retrieval, T5 conversational models. ”What materials were used in constructing the Great Wall of China?”

LLM	LLM Answer	RAG Answer
t5-base_220M	None	stone, brick, tamped earth, and wood
t5-large_770M	False	stone, brick, tamped earth, and wood

Table 10. Fact based question, single paragraph retrieval, causal models. ”How tall is the Pyramid of Giza?”

LLM	LLM Answer	RAG Answer
facebook blenderbot-90M	pickbalfiers fiers predators mutations bender mutations mutations eston mutations eston predators kicker mutations mutations predators mutations pment eston eston eston trout mutations fiers eston eston bender eston eston kicker eston eston fiers eston joreston **end**	design mutpuwillis willis design lease orpotbee willis potpotdistribution fatty fatty composition lease mutfatty fatty willis strip spy lease willis distribution willis strip distribution mutdraft mutpupumutspy **end**
gpt2	The Pyramid of Giza is the tallest building in the world. It is located in the northern part of the country of Egypt. It is one of the tallest buildings in the world	The Great Pyramid of Giza is the tallest man-made structure in the world for over 3,800 years until the construction of Lincoln Cathedral in England in 1311. The Sphinx is
EleutherAI gpt-neo-125M	How tall is the Pyramid of Giza? How tall is the Pyramid of Giza? How tall is the Pyramid of Giza? How tall is	The Great Pyramid of Giza was the tallest man-made structure in the world for over 3,800 years until the construction of Lincoln Cathedral in England in 1311. The Great
gpt2-medium	The Pyramid of Giza, also known as the Great Pyramid of Giza or Pyramid of the Sun, is one of the world’s largest pyramids. It was built in Egypt	The Pyramid of Giza is the tallest man-made structure in the world for over 3,800 years until the construction of Lincoln Cathedral in England in 1311. The Sphinx,
gpt2-large	The Pyramid of Giza is believed to be the tallest structure in the world. The Pyramid of Giza is located in Egypt. The Pyramid of Giza was built between	The height of the Great Pyramid of Giza is 146.6 meters (481 feet). This is the tallest man-made structure in the world for over 3,800 years. The
EleutherAI gpt-neo-1.3B	This is a question that has been asked many times, but the answer is not always so simple. Some of the tallest buildings in the world are located in the United States, and	The Pyramids of Giza are the only remaining structure of the Seven Wonders of the Ancient World. The Great Pyramid of Giza, also known as the Pyramid of Khuf

Table 11. Fact based question, single paragraph retrieval, question answering models. ”How tall is the Pyramid of Giza?”

LLM	LLM Answer	RAG Answer
distilbert-base-uncased-distilled-squad_66M	how tall is the pyramid of giza? [SEP]	146.6 meters
deepset roberta-base-squad2_125M		146.6 meters (481 feet)
bert-large-uncased-whole-word-masking-finetuned-squad_340M	tall	146.6 meters

Table 12. Fact based question, single paragraph retrieval, T5 conversational models. ”How tall is the Pyramid of Giza?”

LLM	LLM Answer	RAG Answer
t5-base_220M	None	146.6 meters (481 feet)
t5-large_770M	False	146.6 meters

Table 13. List based question, single paragraph retrieval, causal models. ”What materials were used in constructing the Great Wall of China?”

LLM	LLM Answer	RAG Answer
facebook blenderbot-90M	horn ba ba squba ba ba da ba ba bal ba ba phba ba eling ba ba sheet ba ba itba da phth ba ba muba da da audprba da **end**	potpotscistrip scipotorscizarpotdistribution scipotstrip strip cupotstrip sciscistrip potsciscibee qpotstrip cuqstrip potqsciqstrip sci__end__
gpt2	There are many materials that were used in the construction of the Great Wall of China, but only a few were used for the construction of the Great Wall of China …	The Great Wall of China was built to protect against invasions from northern tribes. The Great Wall of China is made of various materials, including stone, brick, tamped earth …
EleutherAI gpt-neo-125M	The Great Wall of China was built by the Chinese government during the reign of the Qing dynasty …	The Great Wall of China is over 13,000 miles long. Construction of the Great Wall of China began in the 7th century BC. The Great Wall of China was …
gpt2-medium	The Great Wall of China was constructed by a team of Chinese engineers, engineers, and architects …	The Great Wall of China was constructed of various materials, including stone, brick, tamped earth, and wood. The height of the Great Wall of China varies, with the tallest …
gpt2-large	The Great Wall of China was constructed by the Han Dynasty (206 BC – 220 AD) …	The Great Wall of China is made of various materials, including stone, brick, tamped earth, and wood. The height of the Great Wall of China varies, with the tallest sections …
EleutherAI gpt-neo-1.3B	The Great Wall of China is one of the most famous examples of ancient Chinese architecture …	The Great Wall of China is made of various materials, including stone, brick, tamped earth, and wood. The height of the Great Wall of China varies, with the tallest …

Table 14. List based question, single paragraph retrieval, question answering models. ”What materials were used in constructing the Great Wall of China?”

LLM	LLM Answer	RAG Answer
distilbert-base-uncased-distilled-squad_66M	what materials were used in constructing the great wall of china? [SEP]	stone, brick, tamped earth, and wood
deepset roberta-base-squad2_125M		stone, brick, tamped earth, and wood
bert-large-uncased-whole-word-masking-finetuned-squad_340M		stone, brick, tamped earth, and wood

Table 15. List based question, single paragraph retrieval, T5 conversational models. ”What materials were used in constructing the Great Wall of China?”

LLM	LLM Answer	RAG Answer
distilbert-base-uncased-distilled-squad_66M	what materials were used in constructing the great wall of china? [SEP]	stone, brick, tamped earth, and wood
deepset roberta-base-squad2_125M		stone, brick, tamped earth, and wood
bert-large-uncased-whole-word-masking-finetuned-squad_340M		stone, brick, tamped earth, and wood

Table 16. Synthesis based question, three paragraph retrieval, causal models. ”Which famous structures, both designed or structurally influenced by Gustave Eiffel?”

LLM	LLM Answer	RAG Answer
facebook blenderbot-90M	etically ication ix belle ication belle ication judgment ication belle le tacication belle conception ication le le belle ication planted le tacdden ication tacication judge ix ication le ication judge belle ication ication le **end**	Error generating text for facebook blenderbot-90M
gpt2	They’re not, but it’s hard not to feel a twinge of nostalgia for them …	The Eiffel Tower is located in Paris, France. The Eiffel Tower was completed in 1889. The Eiffel Tower was painted every seven years to prevent it …
EleutherAI gpt-neo-125M	This article is part of a series of articles that explore Gustave Eiffel’s influence on architecture and design …	The Colosseum was completed in AD 80. The Colosseum was designed by Gustave Eiffel, who also designed the Eiffel Tower.
gpt2-medium	There are many, many, many. Some of the most famous of them are: The Eiffel Tower, Paris …	The Eiffel Tower is located in Paris, France. The Eiffel Tower was completed in 1889. The Eiffel Tower is 324 meters tall. The Eiffel Tower was designed by Gustave Eiffel.
gpt2-large	The Louvre, Paris, France The Eiffel Tower, Paris, France The Eiffel Tower, Paris, France …	The Eiffel Tower is located in Paris, France. The Eiffel Tower is 324 meters tall. The Eiffel Tower was designed by Gustave Eiffel.
EleutherAI gpt-neo-1.3B	The Eiffel Tower in Paris. The Eiffel Tower in Paris. The Eiffel Tower in Paris …	The Eiffel Tower is located in Paris, France. The Eiffel Tower was completed in 1889. The Eiffel Tower is 324 meters tall.

Table 17. Synthesis based question, three paragraph retrieval, question answering models. ”Which famous structures, both designed or structurally influenced by Gustave Eiffel?”

LLM	LLM Answer	RAG Answer
distilbert-base-uncased-distilled-squad_66M	which famous structures, both designed or structurally influenced by Gustave Eiffel? [SEP]	statue of liberty
deepset roberta-base-squad2_125M		
bert-large-uncased-whole-word-masking-finetuned-squad_340M		statue of liberty

Table 18. Synthesis based question, three paragraph retrieval, T5 Models

LLM	LLM Answer	RAG Answer
t5-base_220M	None	The Eiffel Tower
t5-large_770M	False	Eiffel Tower and The Statue of Liberty

Table 19. Experiment 1, Model Performance with Sizes in Parameters

Model Name	Model Type	RAG Answer Quality	LLM Answer Time (s)	RAG Answer Time (s)	Size in Parameters
facebook/ blenderbot-90M	causal_model	Incorrect	1.39	1.63	90M
gpt2	causal_model	Over Information	3.59	2.75	124M
EleutherAI/ gpt-neo-125M	causal_model	Partial	3.54	2.79	125M
gpt2-medium	causal_model	Partial	9.36	6.03	355M
gpt2-large	causal_model	Correct	17.03	10.47	762M
EleutherAI/ gpt-neo-1.3B	causal_model	Partial	20.68	19.14	1.3B
distilbert-base-uncased-distilled-squad	qa_model	Correct	0.22	0.09	66M
deepset/ roberta-base-squad2	qa_model	Correct	0.49	0.17	125M
bert-large-uncased-whole-word-masking-finetuned-squad	qa_model	Correct	1.26	0.45	340M
t5-base	t5_model	Correct	1.17	0.90	220M
t5-large	t5_model	Partial	3.40	1.83	770M

Table 20. Experiment 2, Model Performance with Sizes in Parameters

Model Name	Model Type	RAG Answer Quality	LLM Answer Time (s)	RAG Answer Time (s)	Size in Parameters
facebook/ blenderbot-90M	causal_model	Incorrect	1.39	1.63	90M
gpt2	causal_model	Partial	3.59	2.75	124M
EleutherAI/ gpt-neo-125M	causal_model	Incorrect	3.54	2.79	125M
gpt2-medium	causal_model	Partial	9.36	6.03	355M
gpt2-large	causal_model	Partial	17.03	10.47	762M
EleutherAI/ gpt-neo-1.3B	causal_model	Partial	20.68	19.14	1.3B
distilbert-base-uncased-distilled-squad	qa_model	Partial	0.22	0.09	66M
deepset/ roberta-base-squad2	qa_model	Correct	0.49	0.17	125M
bert-large-uncased-whole-word-masking-finetuned-squad	qa_model	Correct	1.26	0.45	340M
t5-base	t5_model	Correct	1.17	0.90	220M
t5-large	t5_model	Correct	3.40	1.83	770M

Table 21. Experiment 3, Model Performance with Sizes in Parameters

Model Name	Model Type	RAG Answer Quality	LLM Answer Time (s)	RAG Answer Time (s)	Size in Parameters
facebook/ blenderbot-90M	causal_model	Incorrect	1.69	2.20	90M
gpt2	causal_model	Incorrect	3.70	3.67	124M
EleutherAI/ gpt-neo-125M	causal_model	Incorrect	3.68	4.19	125M
gpt2-medium	causal_model	Over Information	8.95	8.65	355M
gpt2-large	causal_model	Over Information	16.52	15.63	762M
EleutherAI/ gpt-neo-1.3B	causal_model	Incorrect	21.88	27.92	1.3B
distilbert-base-uncased-distilled-squad	qa_model	Correct	0.16	0.18	66M
deepset/ roberta-base-squad2	qa_model	Correct	0.32	0.35	125M
bert-large-uncased-whole-word-masking-finetuned-squad	qa_model	Correct	1.29	1.12	340M
t5-base	t5_model	Correct	1.33	1.54	220M
t5-large	t5_model	Correct	3.66	3.38	770M

Table 22. Experiment 4, Model Performance with Sizes in Parameters

Model Name	**Model Type**	**RAG Answer Quality**	**LLM Answer Time (s)**	**RAG Answer Time (s)**	**Size in Parameters**
facebook/ blenderbot-90M	causal_model	Incorrect	2.09	2.78	90M
gpt2	causal_model	Over Information	4.11	4.32	124M
EleutherAI/ gpt-neo-125M	causal_model	Incorrect	4.66	4.51	125M
gpt2-medium	causal_model	Over Information	10.92	10.45	355M
gpt2-large	causal_model	Over Information	17.46	14.16	762M
EleutherAI/ gpt-neo-1.3B	causal_model	Over Information	19.06	22.33	1.3B
distilbert-base-uncased-distilled-squad	qa_model	Correct	0.19	0.17	66M
deepset/ roberta-base-squad2	qa_model	Correct	0.55	0.30	125M
bert-large-uncased-whole-word-masking-finetuned-squad	qa_model	Correct	1.40	0.86	340M
t5-base	t5_model	Correct	1.16	1.41	220M
t5-large	t5_model	Correct	3.59	4.72	770M

Table 23. Experiment 5, Model Performance with Sizes in Parameters

Model Name	**Model Type**	**RAG Answer Quality**	**LLM Answer Time (s)**	**RAG Answer Time (s)**	**Size in Parameters**
facebook/ blenderbot-90M	causal_model	Incorrect	2.19	3.09	90M
gpt2	causal_model	Incorrect	4.96	5.57	124M
EleutherAI/ gpt-neo-125M	causal_model	Incorrect	4.14	5.92	125M
gpt2-medium	causal_model	Incorrect	9.49	13.54	355M
gpt2-large	causal_model	Incorrect	18.67	24.57	762M
EleutherAI/ gpt-neo-1.3B	causal_model	Incorrect	18.41	37.47	1.3B
distilbert-base-uncased-distilled-squad	qa_model	Partial	0.14	0.31	66M
deepset/ roberta-base-squad2	qa_model	Incorrect	0.38	0.71	125M
bert-large-uncased-whole-word-masking-finetuned-squad	qa_model	Partial	0.92	1.70	340M
t5-base	t5_model	Partial	0.99	1.53	220M
t5-large	t5_model	Correct	3.56	5.78	770M