

Document-based question answering

Hachem Betrouni^{1,2}

¹National Polytechnic school of Algiers, Industrial Engineering Department, Data Science and AI, Algiers, hachem.betrouni@g.enp.edu.dz

²BIGmama technology, France, hb@big-mama.io

June 14, 2023

Abstract

In this thesis, we present a comprehensive analysis of various open-source embedding models and large language models (LLMs) for document-based question answering (DBQA). The goal of DBQA is to extract relevant information from a given document and answer user queries in natural language. We evaluate the performance of different embedding models, including BERT, E5, and MiniLM, in combination with open source LLMs such as Vicuna, Falcon, and OpenLlama. Our experimental setup involves a retriever-generator framework, where a retrieval system retrieves the most relevant contexts from a document using embedding models. The conditioned generative system then generates a response to the user query based on the retrieved contexts using LLMs. We benchmark the performance of these systems using the SQuAD dataset. We find that the choice of embedding model significantly impacts the retrieval performance. Moreover, we observe that the choice of LLM also plays a crucial role in generating accurate and informative answers. This comparative analysis provides insights into the strengths and weaknesses of various open-source embedding models and LLMs for DBQA tasks. The findings of this study can guide researchers and practitioners in selecting suitable models for their specific DBQA applications and contribute to the advancement of question-answering systems.

To my parents,
to my family, mentors, friends, enemies, and every person who shaped who I am,
to every rock that broke my bones, to every sea waving on the shores,
to Algiers's seagulls waking me up morning, reminded of Allah's boons,
to Collo's men who rescued me,
to you all I dedicate not this thesis but the years that led to its maketh.

Contents

1	General introduction	6
2	State of play	6
2.1	Introduction	6
2.2	Asset-based consulting	6
2.3	BIGmama technology	7
2.4	Mission	7
2.5	Vision	7
2.6	Unique methodology	8
2.6.1	AI starts with problematization.	8
2.6.2	AI is a tool, not an end-goal	8
2.6.3	Hybridization	8
2.7	Software as a service (SaaS)	8
2.8	Knowledge representation	9
2.8.1	Document-based question answering	9
2.9	Conclusion	10
3	State of the art	11
3.1	Introduction	11
3.1.1	NLP Tasks	11
3.1.2	Embeddings	11
3.1.3	Transformers	12
3.1.4	Similarity measures	13
3.2	Many LLMs	14
3.3	Question answering	15
3.3.1	Visual QA	15
3.3.2	Open-domain QA	15
3.4	Methods for document-based question answering	15
3.4.1	Open-book QA	15
3.4.2	Closed-book QA	17
3.5	Retrieval models	17
3.6	Reader Models	18
4	Implementation and results	19
4.1	Introduction	19
4.2	Description	19
4.3	Evaluation	19
4.4	Implementation details	20
5	Conclusion and perspectives	20
6	Annexes	22
6.1	Code	22
6.2	Experiments	22

List of Figures

1	General structure of this document.	6
2	Iterative process of developing an AI asset in Hyko.	9
3	Overview of open-book question answering approaches.	16
4	Overview of closed-book fine-tuned QA.	17
5	GPT3’s performance on TriviaQA [?]. [?]	18
6	Highlevel overview of the whole system.	19

List of Tables

1	Benchmark : score of each system when tested on 20 samples from SQuAD [?] dataset, and evaluated using GPT-3 [?].	20
---	---	----

1 General introduction

For this BIGmama confined the development of a document-based question answering model to us.

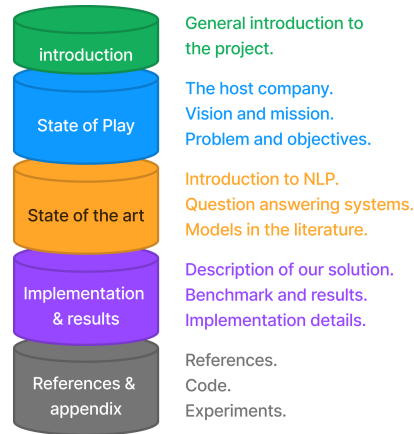


Figure 1: General structure of this document.

2 State of play

2.1 Introduction

Creating value in the market is usually about solving a complex problem or answering a question. When it comes to the Data science and AI sector this used to be tackled with off-the-shelf solutions/applications/answers. However the arrival of LLMs and other fundamental models upset the level of requirements. Creating value now requires expensive, bespoke AI applications. The requirements are no longer to predict a number, but to explain the prediction, it's no longer to recommend a blog title but to generate its whole content.

2.2 Asset-based consulting

This evolvement of the business landscape, driven by rapid technological advancements, changing market dynamics, and increasing competition created a fast-paced environment, where organizations seek innovative ways to optimize their operations, unlock hidden potentials, and achieve sustainable growth.

Today we are well aware that answering these organizations needs for innovation has shifted from writing consulting reports (or any of the traditional approaches) to building tangible assets that concretize the added value. Today, clients are expecting specialised solutions to their problems, rather than the boilerplate solutions consultants traditionally provided.

This has given rise to a thriving industry known as asset-based consulting (ABC).

Asset-based consulting is a specialized field within consulting that focuses on identifying, leveraging, and maximizing the value of a company's assets, AI can be incorporated in the schema of consulting by leveraging the data/knowledge/expertise of the company, incorporating it with intelligent systems, automate tasks and improve the productivity of its users in general.

In the ABC context, assets can include tangible resources such as infrastructure, technology, and inventory, as well as intangible assets like intellectual property, brand equity, and human capital. By harnessing the full potential of these assets, asset-based consultants help organizations drive operational efficiency, enhance performance, and create a competitive advantage.

Unlike traditional consulting approaches that often emphasize external factors and strategies, asset-based consulting takes a more holistic and internal perspective. It recognizes that companies possess unique assets and capabilities that, when strategically managed, or in our case, combined with AI, can serve as the foundation for sustainable competitive advantage and resilience. Here consultants work closely with clients to understand their expertise, identify untapped questions and needs, and design tailored solutions.

Asset-based consulting is known for its multidisciplinary nature, drawing expertise from various fields including finance, operations, technology, marketing, and human resources. Consultants blend their industry knowledge, analytical skills, and business acumen to assess, diagnose, and unlock the hidden value with innovative assets. They often conduct thorough assessments, employ data-driven methodologies, and collaborate closely with expert teams to align the asset usefulness with the overall business objectives.

As industries become increasingly complex and competitive, the demand for asset-based consulting continues to rise. Organizations of all sizes and sectors recognize the need to unlock the value within their assets to stay ahead of the curve. From manufacturing firms seeking to optimize their supply chains to technology companies aiming to capitalize on their intellectual property, asset-based consulting provides a strategic framework and expertise to achieve these objectives.

2.3 BIGmama technology

BIGmama is a French registered startup with an Algerian extension, specialized in data science and AI. They have been developing bespoke predictive applications for more than 8 years (as of the year 2023).

The board of directors counts former CEOs of global groups (Danone, Safran) and the team is assembled from a dozen of high-level data scientists and software engineers.

2.4 Mission

BIGmama's mission is to democratize the access to bespoke AI applications, and to transform companies and individuals by introducing AI in their jobs however possible.

From an original method based on the know-how developed while working with their partners for nearly a decade that we automate or increase with algorithms (Hybrid AI).

What is important at BIGmama today is to be able to industrialize this methodology by proposing a software that encapsulates and hybridizes this expertise with ML models. In this way, this tool will allow them to build state of the art predictive applications, relatively quickly and at an affordable cost.

They believe that a large part of the future of the AI sector will be played out in the ability to articulate human expertise with the countless possibilities offered by AI models.

The last version of the software they are developing (HYKO), should be released by this June.

2.5 Vision

Data science will soon become a commodity, the arrival of large language models (which are considered foundational models in NLP) mark but a start of a new wave of foundational models. In the near future, we can expect the emergence of generalist models that are equivalent to ChatGPT but specialized in computer vision tasks, forecasting, pattern recognition, and other domains.

AI projects that take 6 months in the making, require extensive "hyper-parameter tuning", hundreds of experiments, task specific architectural modeling, and that come at an expensive price will soon become obsolete and outdated.

The future is going to become in the hands of those who are capable of harnessing the power of these generalist models. Algeria and the whole African continent is far behind, and it's no longer a question of closing this gap step by step, a "quantum jump" is required. BIGmama will lead this quantum jump, and will prove that excellence can beam from within the African continent.

2.6 Unique methodology

One of the valuable heritages that BIGmama acquired during the 8 years of actively developing bespoke AI applications to its clients is the unique methodology of work at BIGmama. It's centered around the idea is

2.6.1 AI starts with problematization.

Data science does not start with data science but with a re-framing and problematization work. Our clients most often arrive with a subject (and not problems). A subject in itself is composed of many "hidden" problems (often hidden even to the client), a good consultant at BIGmama knows that this process of problematization (that of decomposing the subject into multiple problems) requires multiple iterations and attempts, and its success is deeply rooted to that of understanding the expertise of the client, their needs and to the skills of the consultant to translate these problems into something addressable by the current AI models.

2.6.2 AI is a tool, not an end-goal

At BIGmama, we firmly believe that AI should be seen as a powerful tool rather than an ultimate goal in itself. While AI technology continues to advance rapidly, its true value lies in its ability to address and solve specific problems or challenges faced by individuals, businesses, and society as a whole. By recognizing AI as a tool, we emphasize its role in enabling and enhancing human capabilities rather than replacing them. And we emphasize that the current approaches in AI are not always the best to solve *every* problem.

2.6.3 Hybridization

We believe that the future of AI lies in what is commonly referred to today as hybrid AI. This is a set of approaches and methodologies aimed at combining the potential of models with purely human knowledge. We believe that this hybridization allows us to put humans at the heart of technological development and to produce tools that are more efficient, easier to maintain, explain and that are far less expensive.

2.7 Software as a service (SaaS)

BIGmama is currently working on a software that intrinsically embeds its mission of democratizing AI and its unique methodology of how to do so. Hyko is an iterative software that helps users formulate their problems properly using AI conversational systems powered by LLMs. It then automatically generates working applications prototypes that combine and orchestrate multiple AI models predisposed in the user's model-base.

Hyko can be described as a three steps iterative process :

Diagnosis and problematization (Scoping)

Automatic prototyping Hyko leverages the ever increasing availability of open-source AI models (Huggingface [?], Github, etc.) and the zero-shot reasoning capabilities [?] of today's LLMs to solve the following tasks :

1. task planning : designing how the tasks should be addressed, setting up the inputs, outputs and the dependencies of each task in the pipeline.
2. model selection : selecting the right model in each step of the pipeline.

Similar to HuggingGPT [?], by solving these tasks Hyko is capable of going from a description of the user's problems to a working AI prototype solution.

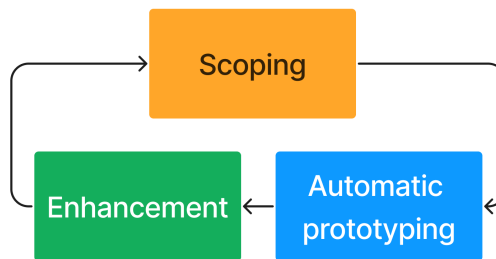


Figure 2: Iterative process of developing an AI asset in Hyko.

Enhancement of the prototype

2.8 Knowledge representation

A very important aspect of the first step (Scoping) is the ability to represent already existing knowledge (technical documents for example) and make it digestible by the LLMs behind the scoping phase.

2.8.1 Document-based question answering

Document-based question answering (DBQA) is a machine learning task that focuses on extracting accurate and relevant answers from a given document and responding in natural language to a specific query.

It involves representing and manipulating content within textual documents, such as articles, technical documentation in order to provide precise answers to user queries.

DBQA systems typically involve a two stages process :

1. context retrieval : employing information retrieval (IR) techniques to extract relating contextual passages from the document.
2. question answering (QA) : using natural language processing (NLP) approaches to answer a query using the retrieved context.

2.9 Conclusion

In this paper we will dive deeper into the different frameworks, models and techniques used in DBQA systems (and similar QA tasks).

In this project we experimented with different DBQA systems, we prepared a small benchmark for few open-source LLMs used (in many combinations) with simple context retrieval systems.

3 State of the art

3.1 Introduction

3.1.1 NLP Tasks

3.1.2 Embeddings

A word embedding is a technique used in NLP to represent words as dense vectors in a high-dimensional space (an embedding). It aims to capture the semantic and syntactic relationships between words, enabling algorithms to process language more effectively.

Word embeddings can be created using two main approaches (or a combination of both): neural and statistical. Neural approaches, use neural networks to learn word representations by capturing semantic and syntactic relationships. Statistical approaches, rely on statistical features of words and their occurrences.

TF-IDF TF-IDF (Term Frequency-Inverse Document Frequency) is a commonly used weighting scheme in information retrieval and text mining tasks, designed to capture the importance of terms in a document collection. It combines two fundamental concepts: term frequency (TF) and inverse document frequency (IDF). TF refers to the number of times a term appears in a document, normalized by the total number of terms in that document. IDF, on the other hand, quantifies the rarity of a term across the entire document collection by taking the logarithm of the inverse of its document frequency.

Document frequency refers to the number of documents in the collection that contain a given term. The TF-IDF weight for a term in a document is obtained by multiplying its TF value by its IDF value. This approach assigns higher weights to terms that appear frequently in a specific document but are relatively rare across the entire collection, effectively capturing their discriminative power. the TF-IDF weight of a term t in a document d can be computed as follows:

$$\begin{aligned} \text{tf}(t, d) &= \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \\ \text{idf}(t, D) &= \log \frac{N}{|\{d \in D : t \in d\}|} \\ \text{tfidf}(t, d, D) &= \text{tf}(t, d) \cdot \text{idf}(t, D) \end{aligned}$$

where $f_{t,d}$ is the raw count of a term in a document, i.e., the number of times that term t occurs in document d , N the total number of documents in the corpus $N = |D|$ and $|\{d \in D : t \in d\}|$ is the number of documents where term t appears.

The resulting TF-IDF weights can be used for various purposes, such as document ranking, text classification, or keyword extraction, enabling the identification of important terms that characterize the content of documents in a collection.

Bert The BERT model [?] is pre-trained on large-scale corpora using unsupervised learning, followed by fine-tuning on specific downstream tasks. The resulting contextual word embeddings capture intricate semantic and syntactic relationships, enabling BERT to excel in various NLP tasks. However BERT [?] uses a cross-encoder (i.e. two sentences are passed to the transformer network and the target value is predicted). This makes it unsuitable for various pair regression tasks due to too many possible combinations (finding most similar pair in a collection of n sentences requires $\frac{n*(n-1)}{2}$ inferences). Sentence-BERT (SBERT) [?], is a modification of the BERT network using siamese and triplet networks, SBERT is able to derive semantically meaningful sentence embeddings (semantically similar sentences are close in vector space).

E5 According to Huggingface embedding models leaderboard [?], the E5 [?] is the current stat-of-the-art (average performance across 62 datasets and more than 7 tasks).

MiniLM MiniLM [?] uses a simple and effective knowledge distillation method to compress large pre-trained Transformer based language models. The student model (MiniLM) is trained by deeply mimicking the teacher’s self-attention [?] modules, which are the vital components of the Transformer networks.

The authors of [?] propose using the self-attention distributions and value relation of the teacher’s last Transformer layer to guide the training of the student, which is effective and flexible for the student models.

The embeddings that come from this model are of high quality, cheap to compute and can be used in a number of down stream tasks.

3.1.3 Transformers

Transformers are a type of deep learning model that have revolutionized natural language processing (NLP) tasks. They were introduced by [?] and have since become the state-of-the-art approach for various NLP applications.

Attention Mechanism:

Attention mechanisms are a key component of transformers. They allow the model to focus on different parts of the input sequence when processing each element. The attention mechanism assigns weights to each element in the input sequence based on its relevance to the current element being processed.

Given an input sequence $\mathbf{X} = \{x_1, x_2, \dots, x_n\}$, the attention mechanism computes a weighted sum of all elements in \mathbf{X} , where the weights are determined by the relevance of each element to the current element. This can be mathematically represented as:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V}$$

where \mathbf{Q} , \mathbf{K} , and \mathbf{V} are the query, key, and value matrices, respectively, and d_k represents the dimension of the key.

Multi-Head Attention:

To capture different types of information, self-attention is performed multiple times in parallel, with different learned linear projections. The outputs of these attention heads are concatenated and linearly transformed:

$$MultiHead(X) = Concat(head_1, head_2, \dots, head_h)W_O$$

where $head_i = SelfAttention(XW_{Qi}, XW_{Ki}, XW_{Vi})$, W_{Qi} , W_{Ki} , and W_{Vi} are learnable weight matrices, and W_O is the output weight matrix.

Encoder Transformer:

In a transformer model, the encoder is responsible for encoding the input sequence. It consists of a stack of identical layers, each comprising a multi-head self-attention mechanism and a feed-forward neural network. The encoder processes the input sequence in parallel and captures the dependencies between different elements.

Let $\mathbf{X} = \{x_1, x_2, \dots, x_n\}$ be the input sequence. The encoder takes \mathbf{X} and produces a sequence of encoded representations $\mathbf{Z} = \{z_1, z_2, \dots, z_n\}$. The encoding process can be expressed as:

$$\mathbf{Z} = Encoder(\mathbf{X}) = EncoderLayer(\dots(EncoderLayer(\mathbf{X})))$$

where $EncoderLayer$ represents a single layer in the encoder, and the ellipsis denotes the stacking of multiple layers.

Encoder-Decoder Architecture:

The encoder-decoder architecture extends the transformer model to tasks such as machine translation, where the input and output sequences have different lengths. The encoder processes the input sequence, while the decoder generates the output sequence.

Let $\mathbf{X} = \{x_1, x_2, \dots, x_m\}$ and $\mathbf{Y} = \{y_1, y_2, \dots, y_n\}$ be the input and output sequences, respectively. The encoder-decoder architecture can be summarized as:

$$\begin{aligned}\mathbf{Z} &= \text{Encoder}(\mathbf{X}) \\ \mathbf{Y}' &= \text{Decoder}(\mathbf{Z}, \mathbf{Y}) = \text{DecoderLayer}(\dots(\text{DecoderLayer}(\mathbf{Z}, \mathbf{Y})))\end{aligned}$$

where *Decoder* represents the decoder, \mathbf{Z} is the encoded representation of \mathbf{X} obtained from the encoder, and \mathbf{Y}' is the predicted output sequence.

The decoder processes the output sequence \mathbf{Y} and attends to the encoded representation \mathbf{Z} to generate the output. The decoder also employs self-attention, allowing it to attend to previously generated elements in the output sequence.

3.1.4 Similarity measures

Similarity measures are used to quantify the degree of similarity or dissimilarity between two objects or vectors. They are widely used in various domains, including information retrieval, recommendation systems, and clustering. Different similarity measures capture different aspects of similarity based on the specific requirements of the task.

Here we mention a few examples of similarity measures commonly used in the literature. However the selection of a suitable similarity measure is contingent upon the characteristics of the data and the particular objectives of the task being performed.

Cosine Similarity:

Cosine similarity is a commonly used similarity measure that computes the cosine of the angle between two vectors. It is particularly useful when comparing the similarity between documents, text embeddings, or high-dimensional data. The cosine similarity between two vectors \mathbf{A} and \mathbf{B} can be calculated as:

$$\text{cosine_similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

where \cdot denotes the dot product and $\|\cdot\|$ represents the Euclidean norm.

Euclidean Distance:

Euclidean distance is a commonly used dissimilarity measure that calculates the straight-line distance between two points in Euclidean space. It is widely used in clustering algorithms, such as k-means. The Euclidean distance between two vectors \mathbf{A} and \mathbf{B} of the same dimension can be computed as:

$$\text{euclidean_distance}(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^n (A_i - B_i)^2}$$

where A_i and B_i represent the i -th elements of vectors \mathbf{A} and \mathbf{B} , respectively.

Jaccard Similarity:

Jaccard similarity [?] is a measure commonly used for comparing the similarity between sets. It is particularly useful in text mining and recommendation systems. The Jaccard similarity between two sets A and B is calculated as the size of their intersection divided by the size of their union:

$$\text{jaccard_similarity}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where $|A|$ and $|B|$ denote the cardinalities of sets A and B , respectively.

Hamming Distance:

Hamming distance [?] is a similarity measure used for comparing binary vectors of the same length. It calculates the number of positions at which the corresponding elements of two vectors are different. The Hamming distance between two binary vectors \mathbf{A} and \mathbf{B} can be computed as:

$$\text{hamming_distance}(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^n (A_i \oplus B_i)$$

where A_i and B_i represent the i -th elements of vectors \mathbf{A} and \mathbf{B} , respectively, and \oplus denotes the bitwise XOR operation.

3.2 Many LLMs

Plethora of large language models (LLMs) are being released week upon week. Some models are made open-source : OpenLlama [?], GPT-2 [?], others are proprietary : GPT-3[?] and GPT-4 [?], some are published for commercial use such as Falcon [?], others are only for research purposes : Llama [?] and Vicuna [?].

Its hard to benchmark the performance of these models due to the wide range of applicable evaluation tasks and the difficulty of filtering out the genuine progress from grandiose claims as well as problems with current QA datasets and unrigorous evaluations [?].

Huggingface recently published an LLM leaderboard [?], where they evaluate publicly available LLMs on 4 key benchmarks :

1. AI2 Reasoning Challenge (25-shot) - a set of grade-school science questions [?].
2. HellaSwag (10-shot) - a test of commonsense inference, which is easy for humans (95%) but challenging for SOTA models [?].
3. MMLU (5-shot) - a test to measure a text model’s multitask accuracy. The test covers 57 tasks including elementary mathematics, US history, computer science, law, and more [?].
4. TruthfulQA (0-shot) - a benchmark to measure whether a language model is truthful in generating answers to questions [?].

Most of the top performing models in this benchmark are either fine-tuned Falcon [?] or fine-tuned Llama [?].

Falcon Large language models are typically trained using a combination of carefully selected high-quality data sources and filtered web data. The curation process aims to create models that perform well across a wide range of tasks. However, as the size of models increases and more training data is required, concerns arise regarding the scalability of the curation process and the availability of unique high-quality data.

The refined web dataset for Falcon llm [?] demonstrates that properly filtered and deduplicated web data alone can yield powerful language models, even outperform state-of-the-art models trained on curated datasets like The Pile [?]. The authors of this dataset[?] have released the Falcon [?] language models with 1.3/7.5 billion parameters trained on this dataset, which serve as valuable resources for this project.

LLaMa The authors of [?] present LLaMA, a collection of foundation language models with parameters ranging from 7 billion to 65 billion. They demonstrate that it is possible to train state-of-the-art models using publicly available datasets alone, without relying on proprietary or inaccessible data. In particular, LLaMA-13B surpasses the performance of GPT-3 (175B) [?] on most benchmarks, and LLaMA-65B is on par with other top models like Chinchilla-70B [?] and PaLM-540B [?]. By making open-sourcing the LLaMA architecture and training code and by making the weights available for researchers, plenty of other models appeared as a result of fine-tuning the original LLaMA models on various tasks and datasets, these fine-tuned models include Alpaca [?] and Vicuna [?].

OpenLLaMa [?] provides a totally open-source replication (weights and code, no research-only restriction) of the original LLaMA [?] models and which surprisingly in some cases outperforms the original models according to their benchmark [?]

3.3 Question answering

3.3.1 Visual QA

3.3.2 Open-domain QA

Open-domain Question Answering (ODQA) is a task examining the ability of models to produce answers to natural language factoid questions drawn from an open set of domains.

DrQA [?] implemented Wikipedia as its knowledge source and this choice has become a default setting for many ODQA studies since then.

However it's relevant in some cases (such as ours) to limit the knowledge source to a signal (or multiple) pre-defined documents, there is a slight difference between the two tasks, to avoid confusion we call the latter "document-based question answering".

Nevertheless approaches in ODQA are applicable for DBQA.

3.4 Methods for document-based question answering

3.4.1 Open-book QA

Open-book models in the field of Open-Domain Question Answering (ODQA) initially retrieve relevant documents then either extract or generate answers based on the information contained in the retrieved documents. We can distinguish mainly two approaches to tackle the ODQA problem :

1. Retriever-reader : this model works toward finding the related context in the documentation than process the retrieved context to **extract** the start/end positions of an answer. The output of the model is the selected context and the identified span of the answer in the context.
2. Retriever-generator : unlike the reader model, the generator model generate free text conditioned with the retrieved context to answer the question.

Retriever-reader Dense Passage Retrieval (DPR) [?], follows a pipeline approach, where documents are retrieved using dense embeddings. These retrieved documents are then passed to a conventional reader-re-ranker, which extracts specific spans of text as answers.

Retriever-generator Retrieval-Augmented Generation [?] is a seq2seq model that jointly learns to retrieve and generate answers. It utilizes dense retrieval and BART [?] for this purpose.

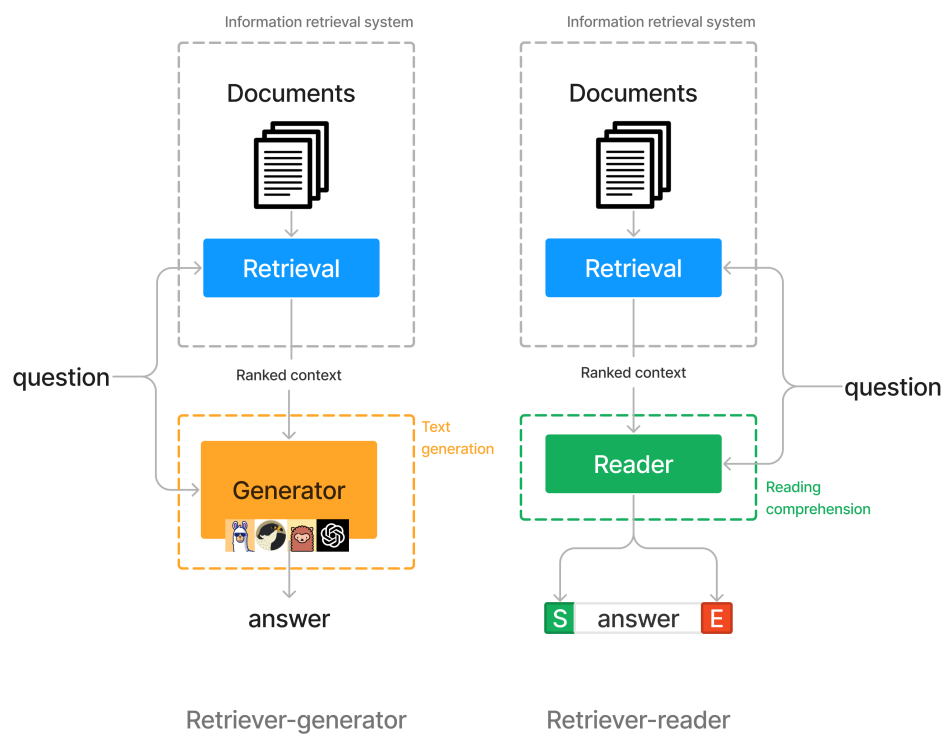


Figure 3: Overview of open-book question answering approaches.

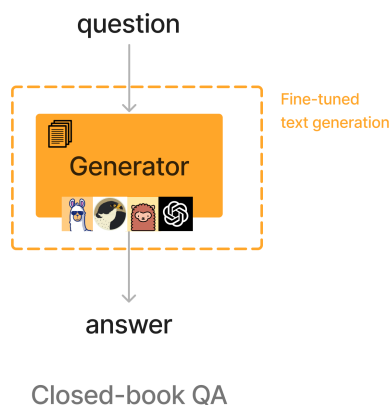


Figure 4: Overview of closed-book fine-tuned QA.

3.4.2 Closed-book QA

Large language models undergo extensive unsupervised pre-training phase using large amount of textual data. With a substantial number of parameters, these models possess the ability to memorize factual information within their weight parameters. Consequently, one is able to employ this property for question-answering tasks without relying on explicit context (shown in figure 4).

The pre-trained language models generate free-form text responses to questions, without explicitly employing reading comprehension techniques. Closed-book models can encode the given documentation within the parameters of the model itself to answer queries, rather than using a retrieval model.

Authors of [?] fine-tune a pre-trained T5 [?] model to answer questions (without access to any external context or knowledge) and were able (at the time) to compete with open-domain systems that explicitly retrieve answers from an external knowledge source when answering questions.

GPT-3 [?] has been evaluated on the closed book question answering task using the TriviaQA dataset [?] without any gradient updates or fine-tuning, the evaluation (figure 5) shows that GPT-3 match/exceed the performance of state of the art (at that time).

3.5 Retrieval models

When it comes to implementing a retriever for a retriever-generator/retriever-reader models, there is mainly two systems :

1. using classic non-learning-based TF-IDF features (“classic IR”).
2. or using dense embedding vectors of text produced by neural networks (“neural IR”).

classic IR For example DrQA [?] adopts an efficient non-learning-based search engine based on bigram hashing and TF-IDF matching.

Another approach used by BERTserini [?] consists of ranking retrieved text segments using BM25 [?], a classic TF-IDF-based retrieval scoring function.

In terms of the effect of text granularity on performance, [?] found that paragraph retrieval ζ sentence retrieval ζ article retrieval.

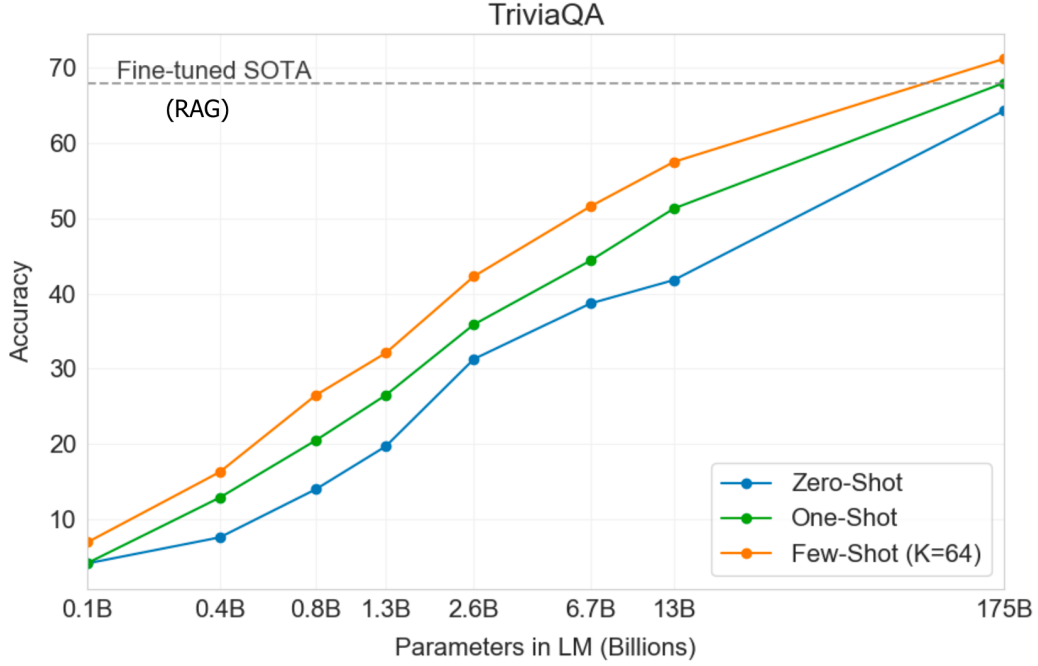


Figure 5: GPT3’s performance on TriviaQA [?]. [?]

Multi-passaged BERT QA model [?] uses elasticSearch with BM25 [?]. They found that splitting articles into passages with the length of 100 words by sliding window brings 4% improvements, and that splitting documents into passages without overlap causes some near-boundary evidence to lose useful contexts thus decreasing the performance.

neural IR Neural IR is a new category of methods for retrieval problems, it mainly uses dense representations of some neural network architectures (e.g. LSTM [?], BERT [?], etc).

After the arrival of many ”general language models” (to do cite llms) many IR for QA systems follow this (or a slightly different) approach :

1. Extract the dense representations of a question and a context passage by feeding them into a language model.
2. Use the dot-product of these two representations as the retrieval score to rank and select most relevant passages.

$$h_q = E_q(q)$$

$$h_z = E_z(z)$$

$$score = h_q^T h_z$$

(however it is not necessary that neural IR out-performs classic IR [?])

3.6 Reader Models

The reader model objective is to learn solve reading comprehension tasks, extract an answer for a question from a retrieved context. Here we only discuss approaches for machine comprehension using neural networks since it yields best performing results.

4 Implementation and results

4.1 Introduction

LlamaIndex [?] bench-marked different indexing methods (different embeddings, LLMs and retrieval systems) following a similar approach to ours. However their focus was on OPENAI proprietary language models (GPT-3 and GPT-4) [?, ?]. Our focus on the other hand was on using open-source embedding models and LLMs that can fully arun on-premise.

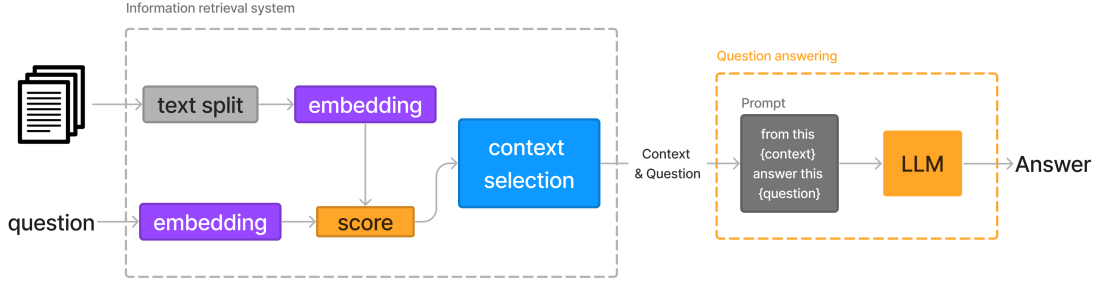


Figure 6: Highlevel overview of the whole system.

4.2 Description

The systems that we experimented with follows the retriever-generator framework, consists mainly of two parts :

a retrieval system that takes as input a document (text) and a query from the user, it splits the document into sentences using a heuristic algorithm developed for the processing the Parallel Corpus for Statistical Machine Translation [?]. Each sentence from the document along the query of the user are going to be embedded by one of the chosen embedding models. We measure the one-to-one similarity (using cosine similarity) between the query embedding and each sentence embedding and we retrieve the top K sentences to be used a context.

a conditioned generative system it takes as input the retrieved contexts and the query to construct a prompt that condition the generation using one of the chosen open-source LLMs. In table 1 we show that we tested all the possible combination of the different embedding models and LMs at hand.

4.3 Evaluation

We bench-marked the different systems on the SQuAD dataset [?]. We sampled 20 with there context and answer (examples can be found in this annex 6.2), for each question and context we tested a different embedding and LLM, each response is them evaluated using GPT-3 [?] (a score of +1 is given if the answer is correct 0 otherwise), results are shown in table 1.

Embedder	Language model		
	Vicuna	Falcon	OpenLlama
Bert	0.4	0.25	0.35
MiniLM	0.35	0.25	0.3
E5	0.35	0.25	0.3

Table 1: Benchmark : score of each system when tested on 20 samples from SQuAD [?] dataset, and evaluated using GPT-3 [?].

4.4 Implementation details

5 Conclusion and perspectives

References

6 Annexes

6.1 Code

6.2 Experiments