Explainable Extractive Generative Multi-hop Question Answering
NLP Course Project & Project Work

Davide Baldelli, Mohammad Reza Ghasemi Madani, Mohammad Amin Nazerzadeh
Master's Degree in Artificial Intelligence, University of Bologna
{ davide.baldelli4, mohammadreza.ghasemi, mohammad.nazerzadeh}@studio.unibo.it

Abstract

In this study, we present a pipeline for an explainable, extractive-generative multi-hop OpenQA model trained on the HotPotQA dataset. Our model features a Retriever-Reader architecture, which consists of an Interaction-based Retriever and Extractor and a Generative module. We show that our model outperforms the baseline model of the dataset. Results show that the Retriever and Extractor submodules perform better on comparison question types rather than bridge question types, while the opposite is true for the Generator submodule. Performance decreases for the Retriever with increasing reasoning difficulty, but not for the Extractor and Generator. Additionally, fine-tuning on the SQuAD v2 dataset was found to enhance multi-hop reasoning ability in the Generator module.

1 Introduction

Question Answering (QA) has been an area extensively studied in Natural Language Processing (NLP) and Information Retrieval (IR). With the advent of deep learning, new methods have emerged to tackle this general problem. In an Open-domain QA setup, answers are extracted from a very large-scale corpus of structured or unstructured data (e.g. the whole internet) (Zhu et al., 2021). For doing so, a variety of techniques are required to be able to compete with human standards and skills in this setup. These techniques operate on different scales for solving this problem. IR techniques can be used to select or rank a smaller set of information from the very large-scale initial corpus while Machine Reading Comprehension (MRC) techniques can be used to generate an answer from a given very small set of provided information. As of this, most OpenQA approaches follow the same pipeline: There exists a Document Retrieval and an Answer Extraction (Zhu et al., 2021). In this project, we tackled this problem on a smaller scale. Instead of dealing with a whole corpus of documents (e.g. whole Wikipedia), we work with a smaller set of documents retrieved from classic IR approaches with high P@k (precision at k) with medium-sized k values (That will cover all the gold documents). By doing so we can take a big leap toward building an OpenQA system.

Another aspect worth noting is that despite the advents of *Retriever-free* approaches (e.g. ChatGPT, GPT3), they lack explainability and scalability to novel data. To cover these restrictions, we chose HotPotQA dataset to simulate an OpenQA setup (Yang et al., 2018). It is a large-scale dataset collected by crowdsourcing based on Wikipedia articles. The answer for each question is obtained by multihop reasoning on *Gold Paragraphs* to test a system's ability to reason with information taken from more than one document. Apart from the provided answer, a set of sentences from which the answer is generated is provided for each question. This gives us the opportunity to test the explainability of our proposed model.

In this project, we propose a Retriever-Reader architecture on HotpotQA. The Retriever selects Gold Paragraphs among a set of relevant and irrelevant Paragraphs. It is a small-bert Interaction-based retriever to gain higher performance compared to representation-based models as will be discussed in the following sections. To pursue the explainability of the Reader module, we took inspiration from extractive-generative QA models. *extractive module* determines sets of sentences in retrieved documents on which reasoning needs to be performed. In the multihop QA it is important to have a model that is able to reason between different documents and understand the relationships between different pieces of information. To do so, we trained a BERT-base model on the Golden Paragraphs sentences to classify them as relative facts to the given query. The extracted facts are then passed to the Generative model to obtain the final answer. The *Generative* module takes the extracted sentences and produces the final answer. The model we have selected for performing this task is *T5-small* from the family of T5 models (Raffel et al., 2020). It is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format.

Following this approach, we outperformed the baseline model introduced in the HotpotQA paper. Along with this result, we observed that the *Retriever, Extractor* model could classify relevant documents for a certain type of question in the dataset better than the other (*comparison* better than *brige*), while the Generator behaves in the opposite way. Our best intuition behind this behavior is that in most of the comparison questions, the entities to be compared are cited in the question, which makes it easier for the *Extractor, Retriever* to locate the relevant spans of information. On the other hand, *comparison* questions often require some mathematical comparison, which is notably hard for Generative models (Cobbe et al., 2021). We also observed that the performance of the *Retriever* decreases with the difficulty of the reasoning process required to answer a question. On the other hand, in the case of the *Extractor*, the f1 score is roughly the same for all levels and the exact match only differs in hard-level questions with a slightly lower score than easy- and medium-level ones while it is not the case for the *Generator* module. This could be due to the fact that there are more medium-level QA pairs in the training data and the level of difficulty is more determined by the difficulty of the supporting facts to retrieve. Finally, We verified the intuition that a *Generator* model fine-tuned on the SQuAD v2 dataset, which is a single-hop QA dataset, can further increase the performance of multi-hop reasoner Reader module, similar to humans behavior as we learn from easy to hard.[1]

2 Background

Question Answering (QA) aims to provide precise answers to the user's questions in natural language. The whole QA landscape can be roughly divided into two parts: textual QA and Knowledge Base (KB)-QA, according to the type of information source where answers are derived from (Zhu et al., 2021). Textual QA deals with unstructured natural language data while KB-QA deals with structured natural language. Specifically, textual QA is studied under two task settings based on the availability of contextual information. **Machine Reading Comprehension** (MRC) aims to empower machines with the ability to read and comprehend specified context passage(s) for answering a given question. While **OpenQA** tries to answer questions without any specified context (Zhu et al., 2021). In fact, MRC can be considered as a step to OpenQA (HARABAGIU et al., 2003).

OpenQA has been studied closely with research in Natural Language Processing (NLP), Information Retrieval (IR), and Information Extraction (IE) (Zhu et al., 2021). Most OpenQA approaches follow a two-module pipeline: A *Document Retrieval* part and a *Answer Extraction* part. In the Document Retrieval phase, systems look for related documents which could hold the whole or part of the answer in them. In the Answer Extraction phase, the answer is extracted from the retrieved documents. With the advents of Deep Learning and MRCs, these systems have evolved to **Retriever-Reader** architectures.

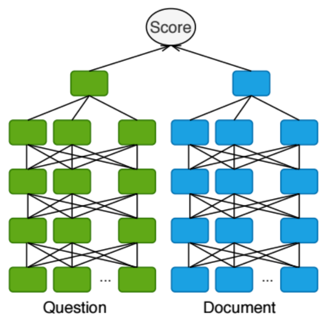
There is a wide variety of approaches for implementing the Retriever and Reader modules. For the Retriever it can be *sparse*, as most classic IR methods such as TF-IDF and BM-25, or it can be dense, which means the calculation and the values are in a continuous vector space. Dense Retrievers can be roughly divided into three types: *Representation-based Retriever*, *Interaction-based Retriever*, and *Representation-Interaction Retriever* (Zhu et al., 2021) as shown in figure 1. In the Representation based, questions and documents are transformed into a common space and based on a similarity score (e.g. cosine similarity), their relevance is determined. ORQA and DPR are two models which adopt this strategy (Lee et al., 2019; Karpukhin et al., 2020). Representation Based methods can be fast since the representation of documents can be precomputed and indexed offline in advance (Zhu et al., 2021). But it can be less powerful because the Representation of the question and the documents are calculated independently, missing the possible interactions between them. On the other hand, Interaction-based retrievers allow for capturing rich interaction between questions and documents as shown in (Nishida et al., 2018). In this setting, a concate-

[1]and can be beneficial for machines too! (Li et al., 2022)

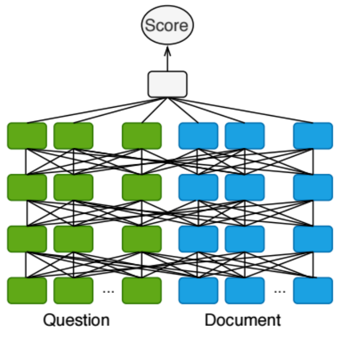
nation of a question and a document is passed to a model to determine if they are relevant or not. The Interaction-based approach required heavy computation and is expensive. The Representation-Interaction approach is a balance between the two previously mentioned approaches to achieve both high performance and efficiency.

Traditionally, the Reader component has been implemented as an extractor, whose task is to predict the starting and ending token of the answer inside a provided context. The classical benchmark for this task has been SQuAD (Rajpurkar et al., 2016), where BERT-like models have achieved SOTA results (Devlin et al., 2018), (Liu et al., 2019). Nonetheless, this approach has evident limitations, as it is not always the case that the answer can be retrieved as a span of words in the context. As a consequence, a Generative module is necessary to build a more general and scalable QA system, and more recently Encoder-Decoder architecture that can achieve RoBERTa results in extractive QA has been proposed, as T5 (Raffel et al., 2020) and BART (Lewis et al., 2019). We have opted for the smallest version of the T5 family for two main reasons: first, it has 60 million parameters, while the smallest version of BART comes with roughly 120 million parameters makes it harder to run and train with our computational availability; furthermore, T5 has been trained on several different downstream tasks that can have been provided it with finer reasoning capabilities.

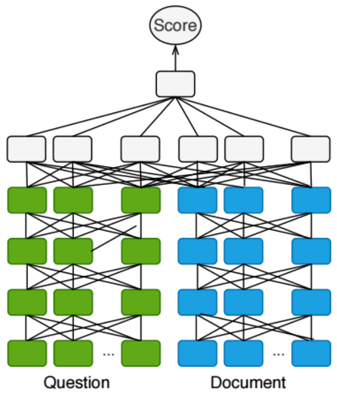
Identifying relevant information among sentences is useful for several applications such as summarization, document provenance, detecting text reuse, evidence retrieval, and novelty detection (Balasubramanian et al., 2007). The task of identifying relevant information in sentences is defined as follows: Given a query sentence, the task is to retrieve sentences from a given collection that express all or some subset of the information present in the query sentence. Figure 4 shows an example of *Supporting Facts* in retrieved documents for a given query. Based on what we proposed so far, one of the key challenges in QA is to retrieve the most relevant sentences from the context that can help to answer the question and it is crucial for extracting the right information from the context, as it helps to reduce the search space and speeds up the overall QA process. There are several techniques used for sentence retrieval in QA, including bag-of-words models, term frequency-inverse document frequency (TF-IDF) models (Thorne et al., 2018), and more recently, deep learning-based models. These techniques vary in their complexity and accuracy, and the choice of a particular technique depends on the requirements of the task and the available resources. In this work, we examine BERT for Sentence Retrieval in our three-stage pipeline.



(a) Representation-Based



(b) Interaction-Based



(c) Representation-Interaction Based

Figure 1: Types of Dense Retrievers (Zhu et al., 2021)

3 System description

Our model consists of three submodules: The Retriever (Document Selector), Extractor, and Generator. Details of each submodule are explained in the following:

3.1 Retriever

The Retriever is a binary sequence classifier based on a pretrained small-bert (Bhargava et al., 2021) architecture. It is a smaller distilled version of Bert-base as introduced in (Sanh et al., 2019). Knowledge distillation is a compression technique in which a compact model - the student - is trained to reproduce the behavior of a larger model - the teacher - or an ensemble of models. To do so, the student is trained with a distillation loss over the soft target probabilities of the teacher. This objective results in a rich training signal by leveraging the full teacher distribution. The model has 4 layers, 512 hidden dimensions, and 8 heads totaling around 29M parameters. It was initialized from *'prajjwal1/bert-tiny'* checkpoints on Huggingface and then fine-tuned on classifying pairs of questions and documents as relevant or irrelevant. The label is determined by the presence of that document in the supporting facts of that question. It is important to note that the number of irrelevant question-document pairs is four times the number of relevant question-document pairs. To compensate for the imbalance in the data, we utilized a weighted loss function proportional to the aforementioned imbalance. Inputs of the model have tokenized question-document pairs truncated or padded with a maximum length of 512 tokens, where the truncation happens only on the document.

3.2 Extractor

The Extractor is in charge of identifying fact sentences from the retrieved documents as a sentence retrieval model. Sentence retrieval is the process of identifying relevant sentences in a text corpus that can be used to answer a question. It is a crucial step in the multi-hop pipeline and determines the quality of the final answer. The quality of sentence retrieval is dependent on the effectiveness of the retrieval method used. Traditionally, the sentence retrieval step has been performed using information retrieval methods, such as the TF-IDF approach (Thorne et al., 2018). However, these methods are not optimized for NLP tasks and do not take into account the semantic meaning of the sentences. With the recent advances in NLP, several neural network-based models have been proposed to perform sentence retrieval in this area. One such model is the BERT-base model, which

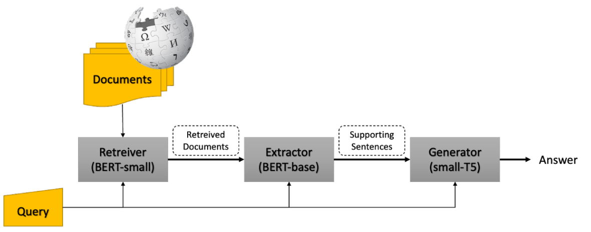


Figure 2: Model pipeline with Retriever, Extractor, and Generator

has shown impressive results in several NLP tasks, including sentence retrieval.

We used the BERT-base pre-trained model to extract the supporting fact sentences in our work. The way we utilize the model in our task is such that we feed the model with pairs of queries and the retrieved documents' sentences and train it to classify them as supporting facts or irrelevant data.

3.3 Generator

Our answer Generator is a text-to-text pre-trained model: T5-small, the tiniest version of the family of T5 models developed by Google (Raffel et al., 2020). T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task, e.g., for translation: `translate English to German:...`, for summarization: `summarize:...`, while for question answering: `question:[...] context:[...]`. T5 has achieved state-of-the-art results on many benchmarks covering summarization, question answering, text classification, and more. In the small version, the feed-forward networks in each block consist of a dense layer with an output dimensionality of 2,048, 8-headed attention, all other sub-layers, and embeddings have a dimensionality of 512, and only 6 layers each in the encoder and decoder. This variant has about 60 million parameters. Following the transfer learning paradigm of pre-training and fine-tuning, we have fine-tuned the model in two steps.

As well pointed in (Li et al., 2022) we pose the question that whether it is too hard and confusing for models to directly learn to answer multi-hop questions from scratch. Therefore, we have started from a version of T5 already trained on the simpler single-hop dataset SQuAD before performing the harder, multi-hop QA task. The model has been initialized from *'mrm8488/t5-small-finetuned-SQuADv2'* checkpoint on Huggingface. We assume that the training on the single-hop dataset has initialized the model with better parameters and make it easier for the model to answer multi-hop questions. In addition, the process of learning from easy to hard is consistent with the widely accepted concept that there is supposed to be a gradual progression of skill acquisition.

We have trained the model using as input texts strings formatted as `question:[...] context:[...]`, where the context is the gold supporting facts, concatenated as if they were a single context, and as target texts the answers.

4 Data

HotpotQA is a dataset for multi-hop question answering. It consists of questions that require reasoning over multiple paragraphs of text to find the answer. The dataset was first introduced in the paper "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering" (Yang et al., 2018) in 2018. The dataset contains 110,000 questions, each with a set of supporting documents and a set of possible answers. The questions cover a wide range of topics and are designed to test a model's ability to perform multi-hop reasoning and understand the relationships between different pieces of information. HotpotQA has been used as a benchmark for evaluating the performance of question-answering models, particularly those that employ multi-hop reasoning. Figure 4 shows an example of a multi-hop question in HotpotQA.

In the HotpotQA dataset, there are two types of questions: Comparison question and Bridge question. The former asks the model to compare two or more entities or concepts in the supporting documents. For example, "Which is taller, Mount Everest or Mount Kilimanjaro?". On the other hand, the latter asks the model to find the relationship between two entities or concepts that are not directly mentioned in the same sentence or paragraph in the supporting documents so the model needs to "bridge" the gap between the two concepts by understanding the relationship between them. For example, "What is the capital of the country where Mount Everest is located?". Both comparison and bridge questions require the model to perform multi-hop reasoning and understand the relationships between different pieces of information in order to find the answer.

Table 1: Evaluation f1 score and exact match of the models

	Validation		Test	
	f1 score	**exact match**	**f1 score**	**exact match**
Retriever	89.72	68.41	88.40	65.48
Extractor	85.81	53.00	83.34	47.63
Generator	86.25	79.80	73.94	60.15

Table 2: Evaluation f1 score and exact match of the **Retriever** model on different question types

	Bridge		Comparison	
	f1 score	**exact match**	**f1 score**	**exact match**
Validation	87.76	62.84	97.77	91.21
Test	86.05	59.06	97.75	91.06

There are also two different settings for HotpotQA, The distractor setting, and the fullwiki setting. In the *distractor* setting of the dataset, for each question, two gold paragraphs are provided along with eight noise paragraphs which are obtained by performing a bigram tf-idf retriever. To also test the *explainability* of model reasonings, they have provided the sentences inside gold documents that human annotators used to derive the answer to each question. The combination of gold paragraphs and gold sentences is called *supporting facts* in the dataset. Further, samples can be categorized into three main subgroups, easy, medium, and hard. The majority of the samples in the train and validation set are medium level but samples in the test set are all hard which means the document retrieval, fact extraction, and answer generation are more challenging than the others. Some statistics of the dataset are shown in figure 3

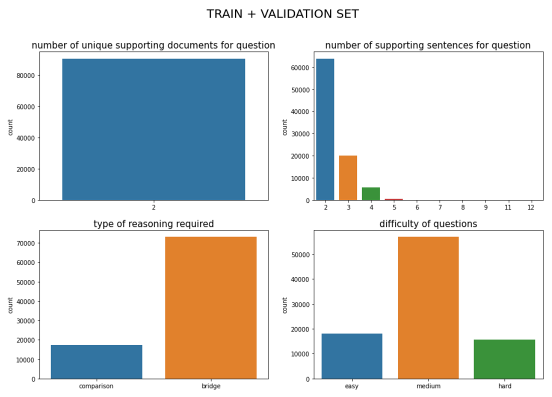


Figure 3: Some statistics of training + validation set in the hotpotQA dataset

5 Experimental setup and results

The Random seed was set to 42 for all the submodules in the pipeline for reproducibility and

Paragraph A, Return to Olympus:
[1] *Return to Olympus is the only album by the alternative rock band Malfunkshun.* [2] *It was released after the band had broken up and after lead singer Andrew Wood (later of Mother Love Bone) had died of a drug overdose in 1990.* [3] Stone Gossard, of Pearl Jam, had compiled the songs and released the album on his label, Loosegroove Records.
Paragraph B, Mother Love Bone:
[4] *Mother Love Bone was an American rock band that formed in Seattle, Washington in 1987.* [5] The band was active from 1987 to 1990. [6] *Frontman Andrew Wood's personality and compositions helped to catapult the group to the top of the burgeoning late 1980s/early 1990s Seattle music scene.* [7] *Wood died only days before the scheduled release of the band's debut album, "Apple", thus ending the group's hopes of success.* [8] The album was finally released a few months later.
Q: What was the former band of the member of Mother Love Bone who died just before the release of "Apple"?
A: Malfunkshun
Supporting facts: 1, 2, 4, 6, 7

Figure 4: An example of the multi-hop questions in HotpotQA. The supporting facts are in *blue italics*, which are also part of the dataset. (Yang et al., 2018)

Table 3: Evaluation f1 score and exact match of the **Extractor** model on different question types

	Bridge		**Comparison**	
	f1 score	**exact match**	**f1 score**	**exact match**
Validation	84.36	48.24	91.77	72.48
Test	81.75	42.63	89.66	67.52

compatibility of validation and test sets among different modules. The definition of metrics is taken from (Yang et al., 2018). We use exact match (EM) and F1 as two evaluation metrics. To assess the explainability of the models they further introduce two sets of metrics involving the supporting facts. The first set focuses on evaluating the supporting facts directly, namely EM and F1 on the set of supporting fact sentences as compared to the gold set. The second the set features joint metrics that combine the evaluation of answers and supporting facts as follows. For each example, given precision and recall of the answer $(P^{(ans)}, R^{(ans)})$ and the supporting facts $(P^{(sup)}, R^{(sup)})$, they calculate join F1 as

$$P^{(joint)} = P^{(ans)}P^{(sup)}, R^{(joint)} = R^{(ans)}R^{(sup)}$$

$$JointF1 = \frac{2P^{(joint)}R^{(joint)}}{P^{(joint)} + R^{(joint)}}$$

joint EM is 1 only if both *ans* and *sup* have an EM of 1 and it is 0 otherwise.

First, we are going to present the performances of the three submodules independently, that is to say, that the input for each module will be the correct one, namely the relevant paragraphs for the Extractor and the actual supporting facts for the Generator. Then we evaluate the final model.

5.1 Retriever

The Retriever model was selected among bert-tiny, bert-mini, bert-small, distilRoberta, and tinyRoberta finetuned on SQuAD v2. dataset. Their respective checkpoint on huggingface is: *prajjwal1/bert-tiny*, *prajjwal1/bert-mini*, *prajjwal1/bert-small*, *distilroberta-base*, and *deepset/tinyroberta-SQuAD2*. One reason for choosing tinyroberta-SQuAD2 was to test its ability to transfer its knowledge from SQuAD v2 to our task. We chose bert-small as our final model as its performance beated the other bert-based models and also roberta based models. You can check different models and their training metrics in the project repository 7.

The model was trained for three epochs. The optimizer was *AdamW* with initial learning rate *lr = 5e-5*. β_1 and $beta_2$ parameters of AdamW were set to 0.9 and 0.999 respectively. A linear decay scheduler was utilized. The training was performed on half-precision floating point for the sake of GPU availability. We used huggingface's Trainer API for training the model. The performance of the final model is described in table 1. We further analyzed the performance of our model on different question types as shown in table 2 The model can easily classify relevant documents for *comparison* type questions but it performs worse on *bridge* type questions. This could be due to the fact that in *bridge* questions, the relevance of a document to the question, is dependant also on other questions as there is an ordered sequence of chain reasoning on documents that need to be done. As an example in the question "What is the

Table 4: Evaluation f1 score and exact match of each model on different question levels in **validation** set

	Easy		**Medium**		**Hard**	
	f1 score	**exact match**	**f1 score**	**exact match**	**f1 score**	**exact match**
Retriever	92.02	75.03	89.49	67.41	87.95	64.52
Extractor	85.56	55.53	86.63	53.84	83.14	47.05
Generator	83.87	74.64	90.43	86.82	73.74	59.95

capital of the country where Mount Everest is located?", you need to first extract the country where Mount Everest is located and then look for the capital of it with the new information. We also evaluated the performance of our model on different question levels on the validation dataset as shown in table 4. As expected, both the f1 score and the exact match of the model decrease with the difficulty of the question.

5.2 Extractor

In the training process, we used the default training argument from the huggingface to train our BERT-base model. In this setting, the utilized optimizer is *AdamW* with linear decay scheduler and it is utilized with initial learning rate *lr = 5e-5*. Besides, β_1 and β_2 parameters of AdamW are set to 0.9 and 0.999 respectively. We implemented a custom trainer with cross-entropy loss for the outputs and applied half-precision floating points. The data imbalance also exists in sentence-level data and we solved it with the same approach we did in Retriever by weighting the predicted labels based on their distribution over the whole training set in our custom trainer. In the end, we applied three epochs to train the model. The important note here is that the training part is done without considering the whole document sentences, which means that we generate training samples by concatenating every single sentence in each of the retrieved contexts and their corresponding queries as a single sample. The labels are boolean values indicating whether the sentence in the sample is a supporting fact for the given query or not. Notice that the final evaluation of the validation and test data is document-based in order to have compatible metrics with other methods on the HotpotQA leaderboard. For instance, the Exact Match (EM) metric is defined such that if the model identifies all the supporting fact sentences in documents and discards all the others, it would be 1 otherwise it is equal to zero. The final EM over the validation/test set is the mean of EMs for each document.

We examined the extractor with bert-tiny, bert-small, and BERT-base. Among the models above, bert-tiny showed poor results compare to others. On the other hand, bert-small and BERT-base performance were roughly similar with slightly better behavior in BERT-base. We chose BERT-base as the final model for the extractor but using bert-small also seems a reasonable alternative

Table 5: Evaluation f1 score and exact match of T5 and T5 trained on SQuAD before and after fine-tuning on HopPotQA on Validation set

	'T5-small'		*'mrm8488/t5-small-finetuned-SQuADv2'*	
	f1 score	**exact match**	**f1 score**	**exact match**
Before fine-tuning	66.11	57.03	55.01	48.06
After fine-tuning	85.78	79.51	86.18	79.75

with much less training time and slightly lower performance. Besides, the BERT-base model has a hidden size of 768 and uses a multi-head attention mechanism with 12 attention heads, allowing it to attend to different representations of the input at the same time and capture complex relationships between words. BERT-base has approximately 110 million parameters which are crucial to its ability to perform well on NLP tasks. The combination of its architectural features makes BERT-base a powerful tool for natural language processing tasks, particularly those that involve sentence classification and sentence retrieval.

In the evaluation process we first evaluate our model on the HotpotQA validation and test splits to see how it performs when it receives all the true documents as can be seen in table 1. Then we perform the evaluation on the outputs produced by Retriever to see how much the performance decays if the model's inputs are not perfect (table 8). Further, we get the performance result on each of the level's categories and question types (bridge and comparison) independently to have an intuition on the dependency of the output on the input data properties. Tables 1, 3, and 4 show the evaluation results of the extractor from different points of view. Also, table 8 illustrates the results when the extractor is in the pipeline together with the retriever and generator. The results show that Comparison question types are easier than the Bridge (table 3). The intuition can be that in the bridge questions, the model has to reason between different sentences and find relationships among them while in the comparison it needs to find sentences for each part element of comparison without needing for reasoning between different sentences. Looking at the difficulty levels of the questions our retriever performs almost uniformly regarding the f1 score and exact match. However, the f1 score and exact match moderately decrease when it comes to hard questions.

5.3 Generator

The model was trained for three epochs. The optimizer was *AdamW* with initial learning rate

Table 6: Evaluation f1 score and exact match of the **Generator** model on different question types

	Bridge		**Comparison**	
	f1 score	**exact match**	**f1 score**	**exact match**
Validation	88.75	81.60	75.78	72.08
Test	76.97	61.24	61.91	55.83

lr = 5e-5. β_1 and $beta_2$ parameters of AdamW were set to 0.9 and 0.999 respectively. A linear decay scheduler was utilized. The training was performed on half-precision floating point for the sake of GPU availability. The performance is described in table 1. To verify our intuition of selecting a model already trained on SQuAD, we have imported the original *'T5-small'* checkpoint and the *'mrm8488/t5-small-finetuned-SQuADv2'* version on the validation set, and evaluated them before and after completing the fine-tuning on HotPotQA, as shown in table 5. We further analyzed the performance of our model on different question types as shown in table 6. It is interesting to note that *'T5-small'* performs better than the SQuAD-trained version before the fine-tuning phase. This behavior could be explained by the fact that the model trained on SQuAD has overfitted on extractive one-hop Question Answering. As a consequence, it expects to find the answer as a span of words in the context, which often is not the case in HotPotQA. While *'T5-small'* is more capable of generalizing to new use cases. Nonetheless, even if by a small amount, *'mrm8488/t5-small-finetuned-SQuADv2'* overperforms the *'T5-small'* after fine-tuning: it seems that our intuition was correct.

We have performed Hyper-Parameters tuning on the generation strategy. In particular, we have tried both a greedy decoding approach and a sampling approach. In greedy decoding, the model selects the token of the highest probability at each step. The highest probability is estimated by keeping track of the k best sequences (as known as beams), at each step selecting the one with the highest probability, continuing the k sequences in the k most probable ways, and out of the k^2 sequences, keep the k most probable. We have tried different values of k, namely $3, 5, 10,$, and 30. On the other hand, with sampling methods, instead of selecting the token with the highest probability, we randomly sample from the probability distribution computed by the language model, and to avoid senseless tokens, we sample between the n most likely tokens. For this approach we have

Table 7: Hyper parameters tuning on decoding strategy for the **Generator**. With **k** and **n** we mean respectively the number of beams and the size of the subset where to sample.

Greedy decoding

k	**f1**
3	86.17
5	86.07
10	86.08
30	86.08

Sampling decoding

k	**n**	**f1**
3	10	86.46
3	30	86.37
10	10	86.45
10	30	86.39
20	10	86.43
20	30	86.41

Table 8: Final Evaluations on Test set

	ANS		**SUP**		**JOINT**	
	f1 score	**exact match**	**f1 score**	**exact match**	**f1 score**	**exact match**
Our model	60.61	48.28	74.68	34.75	48.93	20.78
Baseline model	59.02	45.60	64.49	20.23	40.16	10.83

tried all the combinations with $k \in \{3, 10, 20\}$ and $n \in \{10, 30\}$. To reduce the computational time, we have computed the tuning on a subset of 2000 samples of the validation set. Even if the variations in performance are slight, the model performs better with $k = 3$ and $n = 10$, as shown in table 7.

5.4 Final Results

Lastly, we evaluate the whole pipeline's performance on the metrics mentioned earlier. We compare our results with the baseline model of the dataset (Yang et al., 2018) as shown in 8. Our model outperforms the baseline model by 8.9 points on the joint F1 score and 10 points on the joint EM. We have further studied the performance of our model on different question types, as shown in 9. We can see that the difficulty of the Generator to reason mathematically is compensated by the facility with which the supporting facts for comparison questions are retrieved.

6 Conclusion

In this project, we built a pipeline for an explainable extractive generative multi-hop OpenQA model. It was trained on HotPotQA dataset and was based on a Retriever-Reader architecture.

Table 9: Evaluation of the final model on different question types

	Bridge		Comparison	
	f1 score	**exact match**	**f1 score**	**exact match**
ANS	61.66	47.81	56.22	50.37
SUP	71.41	27.73	88.05	63.01

We built an *Interaction-based* Retriever and Extractor to gain higher performance compared to the *Representation-based* model and the baseline model of the dataset. We observed that these two submodules can perform better on *comparison* type questions compared to *bridge* type questions. While in the Generator submodule, the opposite was observed. We also observed that the performance of the Retriever decreases with the difficulty of the reasoning process required to answer a question, while it is not the case for the Extractor and the Generator modules. Finally, we observed that a Generator model finetuned on SQuAD v2 can perform better on multi-hop reasoning due to having the previous knowledge of one-hop reasoning and being able to transfer it to the new context.

A promising direction for further analysis is what is observed in figure 5. One can observe there are two main regimes of the bivariate distribution of $F1^{(ans)}$ and $F1^{(sup)}$. The top one (which is skewed to top right) predicts a positive correlation between $F1^{(ans)}$ and $F1^{(sup)}$ - as one can expect the performance on generating an answer can increase when the performance of retrieval and extraction increases. While the bottom regime includes the samples in which the Reader module fails independently of how the Retriever and the extractor models perform. We think studying the properties of these samples can reveal interesting findings in the following studies.

7 Links to external resources

- Link to the hotpotqa dataset on huggingface
- Link to our huggingface repository. Models of submodules and the output of different submodules can be found here

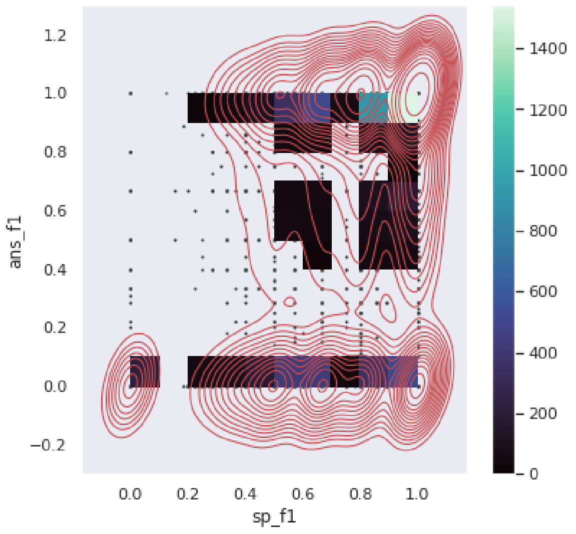


Figure 5: Bivariate distribution of $F1^{(ans)}$ and $F1^{(sup)}$. The colored tiles correspond to the density of samples in that area.

References

Niranjan Balasubramanian, James Allan, and W. Bruce Croft. 2007. A comparison of sentence retrieval techniques. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, page 813–814, New York, NY, USA. Association for Computing Machinery.

Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. 2021. Generalization in nli: Ways (not) to go beyond simple heuristics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

SANDA M. HARABAGIU, STEVEN J. MAIORANO, and MARIUS A. PAŞCA. 2003. Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *CoRR*, abs/1906.00300.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Xin-Yi Li, Wei-Jun Lei, and Yu-Bin Yang. 2022. From easy to hard: Two-stage selector and reader for multi-hop question answering.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2018. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. *CoRR*, abs/1808.10628.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. *CoRR*, abs/1803.05355.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *CoRR*, abs/2101.00774.