Question Answering over Wikipedia Passages

Gioele Zerini
g.zerini1@studenti.unipi.it

Matilde Bisi
m.bisi@studenti.unipi.it

Niccolò Maestripieri
n.maestripieri@studenti.unipi.it

Irene Poli
i.poli5@studenti.unipi.it

Abstract

This project aims to implement an extractive Question Answering (QA) system utilizing Wikipedia passages as the information source. Given a question and a relevant paragraph (i.e., the context) containing the answer, the model is expected to accurately identify and return the text span most likely to contain the answer. We primarily employed BERT-base models, recognized for their effectiveness in context analysis and their robustness as a baseline in the field. Our methodology leveraged the Hugging Face library, utilizing "fast" tokenizers to optimize performance and the accuracy of offset mapping. The experimental analysis involved fine-tuning BERT, ALBERT, and SpanBERT on the SQuAD 1.1 and SQuAD 2.0 datasets. We developed a preprocessing phase to handle tokenization and context chunking, and a postprocessing phase to analyze model predictions, converting logits into text spans. For SQuAD 2.0, we implemented specific handling for unanswerable questions by comparing the best span's score with that of the [CLS] token to determine a "no-answer" prediction. Our evaluation functions for SQuAD 2.0 were custom-implemented due to issues with standard functions not properly recognizing our method for marking unanswerable questions. The results obtained on SQuAD 1.1 and SQuAD 2.0 are largely consistent with existing literature, despite minor variations attributable to the use of "base" model configurations (e.g., for SpanBERT, which was "base" instead of "large" in reference papers) and the specific dataset splitting (less training data due to split). ALBERT and SpanBERT consistently demonstrated superior performance compared to BERT, confirming the effectiveness of these more recent architectures. This study underscores the continued relevance of transformer-based models for extractive QA and provides comparative insights into their performance under various dataset conditions.

1 Introduction

Question Answering (QA) is an interesting subfield of Natural Language Processing that aims to provide precise, concise, and accurate answers to questions expressed in natural language. Unlike traditional search engines, which return a ranked list of relevant documents, QA systems seek to deliver the answer directly as a few words or a passage, permitting users to save the time needed to read all the documents and identify the answer.

QA systems can be categorized into extractive and generative approaches. Generative QA consists in generating the answer in natural language, permitting a good flexibility in terms of summarization, reinterpretation, and multi-step reasoning. These models often rely on generative architectures or hybrid approaches such as Retrieval-Augmented Generation (RAG). While powerful, they can suffer from issues such as hallucination, where the generated content is plausible-sounding but factually incorrect.

HLT final project report (2025).

In extractive QA, instead, the answer is directly taken from a context, which could be a small text or a paragraph. In this case the task is more similar to the span detection or machine reading task, since the model already has the passage containing the answer, and the aim is to detect the most relevant part. This approach is less flexible since the answer must be explicitly present in the text, but has the advantage of being more accurate and precise if this requirement is satisfied. Therefore, extractive QA can be an excellent approach in case of simple questions with explicit answers.

In this project, our goal is to implement an extractive QA system using Wikipedia passages as the source of information. Given a question and a relevant paragraph containing the answer (context), the model is expected to accurately identify and return the span from the passage that is most likely to contain the answer.

We use mainly BERT-base models because they represent a strong baseline in the field, thanks to their ability to analyze context effectively. Moreover, their wide usage allows us to compare our results with previous experiments to validate our observations. We aim to analyze and compare the performance of different models, starting from more general and simple architectures and then moving to more complex and specialized ones, in order to evaluate the improvements.

In the next section we present the main models and datasets used in the literature and provide a brief review of some papers in the field. In sections 3 and 4 we describe our implementation and experiments, while in sections 5 and 6 we discuss and evaluate the results.

2 Background

Given the extensive coverage of extractive question answering in recent years, this section is divided into three parts. We will first present the models used to solve this task, followed by a discussion of the datasets employed. Finally, we will delve deeper into related papers that address the same task as ours.

A. *Models*

Our primary model of interest for this task is BERT [Devlin et al., 2019], a transformer-based architecture known for its versatility. BERT can be readily fine-tuned across numerous NLP tasks, including extractive question answering. This fine-tuning process is straightforward, requiring only the addition of an output layer and subsequent training on new data—a widely adopted methodology within the field. Furthermore, upon its initial release, BERT achieved state-of-the-art performance in extractive question answering. Consequently, we've selected BERT as our baseline model for evaluating our experimental results.

The advent of BERT paved the way for the development of many BERT-based models, some of which have demonstrated superior performance in extractive question answering. One such model is SpanBERT [Joshi et al., 2020], specifically designed to improve the representation and prediction of text spans. This capability is highly advantageous for our task, which requires identifying a precise segment of text as the answer within a given context. Other relevant BERT-based models frequently employed in this field include RoBERTa [Liu et al., 2019], recognized for its enhanced pre-training, and AlBERT [Lan et al., 2020], a more computationally efficient variant of BERT, making it suitable for environments with limited resources.

Other BERT-inspired models are also employed for this task, such as XLNet [Yang et al., 2019], which utilizes an autoregressive pretraining approach, and ConvBERT [Jiang et al., 2020], notable for its distinct attention mechanism. Finally, BART [Lewis et al., 2020], while primarily designed for generative tasks, has shown good results in extractive question answering. The emergence of transformer-based models like BERT has largely superseded older approaches based on LSTMs and RNNs, which are predominantly found in papers dating back to 2016 [Chen et al., 2017].

B. *Datasets*

Regarding the datasets used for extractive question answering, SQuAD [Rajpurkar et al., 2016] (in both its 1.1 and 2.0 versions) stands out as the most common baseline in the literature. We've chosen SQuAD as our baseline due to its suitability for our task; it's composed of (question, answer, context) tuples, where the context is a Wikipedia paragraph and the answer is a text span extracted directly from it. The dataset is publicly available and user-friendly, containing over 100,000 question-answer pairs across more than 500 Wikipedia paragraphs. In 2018, two years after the initial release, SQuAD's

authors introduced a new version [Rajpurkar et al., 2018] that includes unanswerable questions. For these, no direct answer exists within the context, but a plausible distracter is present to challenge the model. The addition of over 50,000 unanswerable questions significantly increases task difficulty, as some models on the first SQuAD version relied solely on selecting a context span related to the question.

Beyond SQuAD, other datasets are also employed for extractive question answering. Natural Questions (NQ) [Kwiatkowski et al., 2019] is particularly suitable for our task, as its questions are derived from full Wikipedia articles rather than short paragraphs, unlike SQuAD. Additionally, NQ includes both short and long answers when available. These nuances make NQ more challenging to use than SQuAD. Other Wikipedia-sourced datasets include HotPotQA [Yang et al., 2018], which favor multi-hop reasoning, QuAC [Choi et al., 2018], which features conversational information, and WikiQA [Yang et al., 2015], a smaller dataset predating SQuAD's release. Extractive question answering is also performed using datasets not sourced from Wikipedia. Examples include NewsQA [Trischler et al., 2017] and CovidQA [Möller et al., 2020], which, similar to SQuAD, collect question-answer pairs but utilize CNN articles and COVID-19 related articles, respectively, as their source material. Other notable datasets are TriviaQA [Joshi et al., 2017] and SearchQA [Dunn et al., 2017], both of which gather questions from web-based trivia quizzes or TV shows.

C. *Related Works*

The extractive QA problem has been addressed in different ways by several works, including very recent ones, and some issues are still open and relevant.

Before the introduction of transformers, the most commonly used approaches were based on RNNs. For example, [Chen et al., 2017] built a system capable of answering questions using Wikipedia as the knowledge source. Their model is composed of two main parts: the first retrieves relevant documents from the entire corpus (document retriever), and the second analyzes these documents to extract the answer. This second part corresponds to the task we focus on. To perform it, the relevant paragraph and the question are encoded and passed through two 3-layer bidirectional LSTM networks. The question vectors are then combined into a single vector. All vectors are passed to two classifiers that compute the probability for each token of being the start or the end of the answer span. The span with the highest probability is selected as the final answer.

Since 2018, transformers have been widely used for many NLP tasks, including extractive QA. [Pearce et al., 2021] focused on comparing the performance of several BERT-based models on different datasets to evaluate their complexity and the models' generalization capabilities. The datasets used are NewsQA, SQuAD 2.0, QuAC, and CovidQA, while the models fine-tuned include XLNet, BERT, RoBERTa, AlBERT, ConvBERT, and BART. The results are presented by analyzing separately the performance on different datasets and models. Regarding the datasets, SQuAD achieved the highest scores, probably because it contains short and purely extractive answers, which makes it particularly suitable for this kind of task. The second-best dataset is NewsQA, followed by CovidQA, which is a small dataset with very long questions, and QuAC, which also includes open-ended questions. These results supported our choice to use SQuAD, as they show that it is the most appropriate dataset for the task, and therefore the most suitable for an initial analysis. Looking at the models, RoBERTa and BART achieved the best performance. RoBERTa is a larger and more complex model, while BART has an encoder-decoder architecture that is especially effective for context understanding and token classification.

Another interesting and very recent work was proposed by [Thottingal, 2025]. In this case, the answer extraction is not based on short spans, but rather on full paragraphs: the goal is to extract from Wikipedia a small paragraph that contains the answer. Traditional approaches use similarity scores, such as cosine similarity between the embeddings of the question and of the text chunks. However, this often leads to low scores even when the passage is relevant, due to the different semantic structure of the questions (interrogative) and the text (declarative). As an alternative, the author proposes using a LLM to generate a set of questions related to each paragraph, and then comparing the original query with these generated questions instead of with the raw text. The results show that this strategy produces higher similarity scores and helps improve the retrieval of relevant passages.

3 Methodologies

In this section, we present the methodologies used in our work. We primarily relied on the Hugging Face library, as it implements the models we planned to use for our task. We considered various models, all transformer-based, to compare their results with those from reference papers. Furthermore, we implemented the preprocessing to set up the data for the models and the postprocessing to evaluate the obtained results.

3.1 Models

For our models, we began with BERT as our baseline, specifically using the "BertForQuestionAnswering" configuration of the "bert-base-uncased" model from the Hugging Face Transformers library. We opted for the "bert-base" model over "bert-large" due to its significantly fewer parameters (around 110M versus 340M). This decision was driven by limited memory capacity and the desire to minimize training time, which remains substantial even with "bert-base." A key characteristic of this model is its span classification head—a linear layer atop the hidden states output—which computes span start and end logits, representing the probabilities that a text segment is the beginning or end of the answer to a given question.

Similar tools were used for ALBERT, as the Hugging Face Transformers library also provides a question-answering version for this model. However, for SpanBERT, we utilized the "AutoModelForQuestionAnswering" class, as SpanBERT isn't directly included in the core Transformers library. This class simply loads a model from Hugging Face by its name, and we, of course, selected a SpanBERT model available there. For both ALBERT and SpanBERT, we chose the base versions for the same reasons outlined for BERT.

All models available in the Hugging Face library are pretrained, requiring only fine-tuning on the desired datasets. This is a standard NLP methodology, with numerous libraries offering tools to facilitate the process. We fine-tuned these models on the datasets selected for our project: SQuAD 1.1 and SQuAD 2.0, which we will discuss in the next section.

3.2 Preprocessing

The preprocessing stage involves preparing the raw data from the dataset for model consumption. Since all our models are BERT-based, the implementation is largely consistent, though it varies slightly between datasets due to the need to handle unanswerable questions in SQuAD 2.0. The initial step in this process is tokenization. For this, we utilize the tokenizer classes provided by the Hugging Face Transformers library: "BertTokenizerFast" for BERT and "AlbertTokenizerFast" for ALBERT. For SpanBERT, we employ the "AutoTokenizer" class for the same reasons outlined previously, ensuring the use of its "fast" version. We chose these "fast" tokenizer versions not only for their superior performance and implementation but also for crucial features like offset mapping, which are vital for our extractive question answering task and are unavailable in the base versions. The tokenizer allows us to distinguish the question from the context and to split the context into chunks when it's too long for the model to handle. It also adds special tokens like [CLS] and [SEP], which aid the model's understanding of the data.

Following tokenization, we proceed to the mapping stage, the final step before model training. In this stage, we calculate the answer's position in terms of tokens, converting it from its character-based position. Once this process is complete, the dataset is ready for the training phase, possessing all the necessary features the model requires. The mapping phase necessitates distinct functions for training and testing. In the test phase, we don't need the answer's start and end token positions, as the model's objective is to find them. Instead, we must associate the various context chunks with unique IDs, enabling us to reassemble the full text later. Finally, we address the matter of unanswerable questions. To mark such questions, we set the answer's position to the [CLS] token at the beginning of the question-answer pair. This is a common solution found in existing literature. To maintain consistency, context chunks from answerable questions that do not contain the actual answer are also marked with the [CLS] token's position.

3.3 Postprocessing

The postprocessing stage involves analyzing our model's predictions on the test set to retrieve the final results. Models output predictions as start and end logits, which represent the probabilities for each token in the context to be the beginning or end of the answer span. Using these probabilities, we must calculate which text span is most likely to be the answer. To do this, we sum the start and end logits for every possible span, up to a maximum length of 30 tokens (a commonly used value in existing literature). We then select the span with the highest summed value. Crucially, we only consider summing logits where the start token's position precedes the end token's position, thereby excluding inadmissible spans.

In the case of SQuAD 2.0, which includes unanswerable questions, we also need to account for "no-answer" predictions. This is done by comparing the score of the best candidate span with the score assigned to the [CLS] token (used to indicate no-answer). Specifically, we compute the difference between the sum of the logits for the [CLS] token and the sum of the logits for the best span. If this difference exceeds a certain threshold, we classify the prediction as "no-answer." To optimize performance, we evaluate thresholds in the range (-2.0, 2.0) with a step size of 0.1, selecting the value that yields the highest F1 score on the validation set.

After determining the most likely answer span in tokens, we convert it back to its character-based version. This conversion is necessary because the performance metrics used to evaluate our models require text in character form. We achieve this conversion using the previously mentioned offset mapping, which was constructed during the tokenization phase. Finally, we evaluate our results using the chosen metrics, which will be explained in the next section. For the SQuAD 1.1 dataset, we utilized the metrics already available in the Hugging Face Transformers library, requiring only that we prepare our results in the function's expected format. However, for the SQuAD 2.0 dataset, we encountered an issue where the standard evaluation function did not properly recognize our method for marking unanswerable questions. We resolved this by implementing our own functions to calculate the evaluation metrics.

4 Experimental analysis

4.1 Task(s)

As introduced in the background, the task addressed in this project is extractive question answering over Wikipedia passages. In particular, we aim to evaluate and compare different models in their ability to extract an answer to a given question from a corresponding context paragraph.

For each input example, the model receives a tokenized question and context, which are concatenated and passed through the model. The task consists in predicting the start and end positions of the answer span within the context. After obtaining the start and end logits from the model, the predicted answer is reconstructed by mapping the selected token indices back to the original text.

The models are trained and evaluated on the SQuAD datasets: SQuAD v1.1 and SQuAD v2.0. The main difference between the two is that SQuAD v2.0 includes unanswerable questions; in such cases, the model is expected to predict a "no-answer," which is typically represented by selecting the special [CLS] token as both the start and end positions. Both datasets are publicly available and can be easily accessed through the Hugging Face Datasets library. Since the official test set is not publicly released, we split the original training set into training and validation subsets (90% and 10%, respectively), and used the official validation set as our test set.Each sample in the dataset is represented as a dictionary with the following fields:

- id: a unique identifier for the question;
- title: the title of the Wikipedia article;
- context: the paragraph containing the answer;
- question: the question posed by the annotator;
- answers: a dictionary containing one or more correct answers with fields text (the actual answer string) and answer_start (its start character index in the context).

This task presents several challenges. First, the models must understand the semantic relationship between the question and the context in order to locate the correct answer span. In the case of SQuAD v2.0, the task is even more complex, as the models must also learn to detect when no answer is present in the context. This requires not only deep language understanding but also some capacity to model uncertainty and avoid overconfident predictions.

In our experiments, the tokenized inputs are structured as: [CLS] question [SEP] context [SEP], following the standard format used in transformer-based architectures. The model outputs a score for each token position indicating the likelihood of being the start or end of the answer span. At inference time, the most probable start and end positions are selected, subject to constraints such as maximum answer length. In the case of SQuAD v2.0, the score of the best span is compared against the score of the [CLS] token to determine whether the model should return an empty answer.

This task serves as a benchmark for evaluating and comparing the performance of different transformer-based models in terms of span extraction accuracy and handling of unanswerable questions.

4.2 Experimental settings

In this work, we evaluate each model using its optimal hyperparameter configuration to ensure that performance comparisons are made under the best achievable conditions for each architecture.

To identify the best configuration, we performed a systematic random search over a defined hyperparameter space, including learning rate, batch size, and number of training epochs. The search ranges were chosen based on a combination of values reported in the literature and empirical considerations specific to the SQuAD datasets and the selected models. Each configuration was trained and validated using a 10% validation split from the training set to estimate its generalization ability on unseen data. Typically, we ran 2 to 3 training sessions per configuration, selecting the best-performing model based on validation metrics.

In the case of SpanBERT-base, the situation was slightly different: we did not find explicit recommendations for this specific variant in the literature. As a result, we adopted the hyperparameter ranges suggested for SpanBERT-large, adapting them via random selection to suit our computational budget. While this introduces some uncertainty in the optimization, it also reflects realistic conditions where full grid search is infeasible, and suggests that even without exhaustive tuning, SpanBERT-base can still deliver strong performance.

During training, we continuously monitored key performance metrics such as F1 score and Exact Match (EM) on the validation split. To improve training efficiency and resource usage, we enabled mixed-precision training (fp16=True). Additionally, we applied weight decay regularization (weight_decay=0.01) to reduce overfitting and improve generalization. Checkpoints were saved at the end of each epoch (save_strategy="epoch") to facilitate model inspection and rollback if needed.

Once the best-performing hyperparameter combination was identified, we retrained each model from scratch on the full training set (i.e., combining the original training and validation splits), and then evaluated it on the final blind test set—corresponding to the original SQuAD validation set—which was never seen during any stage of training or model selection.

All training and evaluation procedures were carried out on an NVIDIA Tesla T4 GPU provided by Google Colab, ensuring consistent hardware conditions throughout the experiments

4.3 Results

This section provides an overview of the performance results obtained by our models on the SQuAD v1.1 and SQuAD v2.0 datasets. The evaluation focuses on two standard metrics widely used in extractive question answering:

- Exact Match (EM): the percentage of predictions that match exactly the ground truth answer.
- F1 Score: the harmonic mean of precision and recall at the token level, reflecting partial overlaps between predicted and true answers.

We report the results obtained in our experiments alongside reference values taken from the original publications for each model [Devlin et al., 2019] [Li and Zhang, 2024] [Lan et al., 2020] [Joshi

et al., 2020], when available. In the tables below, ref EM and ref F1 denote the scores reported in the literature, while EM and F1 represent the values obtained by our trained models. An asterisk (*) indicates that the reference values correspond to the large version of the model (SpanBERT-large), while our experiments were conducted with the base version, depending on availability.

Human performance [Rajpurkar et al., 2016] [Rajpurkar et al., 2018], reported in both datasets, serves as an upper bound benchmark. These values were computed by collecting at least two additional answers per question in the development and test sets. In this secondary annotation task, crowdworkers were shown the same question and context (paragraph), and were asked to select the shortest span that correctly answered the question. If no answerable span was found, the annotator could submit the question without selecting an answer. This process reflects inter-annotator agreement and gives an estimate of how well humans agree with each other on the task, rather than an absolute ground truth.

Table 1: Results obtained by the models on SQuAD v1.1, along with reference values reported in the literature. Values marked with * refer to results from the SpanBERT-large model.

Model	EM	F1	ref EM	ref F1
Human			80.30	90.50
BERT	80.13	87.78	80.80	88.50
AlBERT	83.53	90.69	82.30	89.30
SpanBERT	84.21	91.33	88.80*	94.60*

Table 2: Results obtained by the models on SQuAD v2.0, along with reference values reported in the literature. Values marked with * refer to results from the SpanBERT-large model.

Model	EM	F1	ref EM	ref F1
Human			86.30	89.00
BERT	71.14	75.58	71.59	74.72
AlBERT	76.60	80.47	77.10	80.00
SpanBERT	77.91	82.23	85.70*	88.70*

All models were fine-tuned on the respective training splits of SQuAD v1.1 and v2.0. Evaluation was conducted using the official SQuAD validation script available online. As a result, the scores we report may not exactly match the reference values from the literature, which are typically obtained on the original SQuAD test sets, not publicly available.

As expected, model performance is generally higher on SQuAD v1.1 than on v2.0, where the presence of unanswerable questions introduces greater difficulty. Our results closely align with published reference values, with minor differences that can be attributed to factors such as model size (e.g., SpanBERT-base vs. large), reduced training data due to dataset splitting, or improved hyperparameter tuning via validation. These trends and model comparisons will be further explored in the Discussion section.

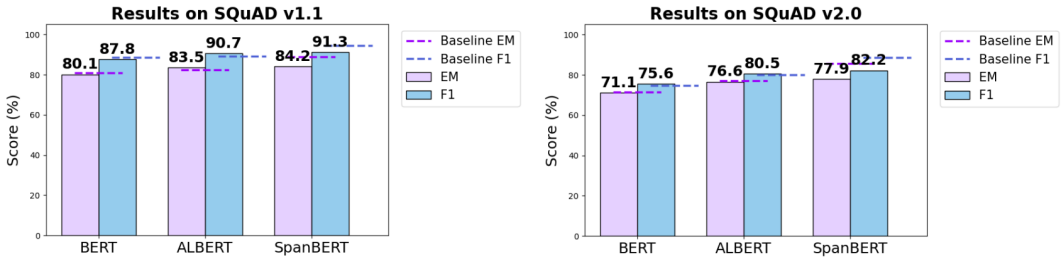

Figure 1: Results of our models on SQuAD v1.1 and SQuAD 2.0 datasets with comparisons to paper results

5 Discussion

The results obtained in our experiments provide several meaningful insights into the current landscape of transformer-based models for extractive question answering over Wikipedia passages.

One of the most relevant findings emerging from our study is that even relatively lightweight transformer models, such as BERT-base, ALBERT-v2, and SpanBERT-base, are capable of delivering strong performance on extractive question answering tasks. Their effectiveness remains remarkable even when compared to larger and more recent architectures like RoBERTa or BART, which are known to be more computationally demanding. This observation reinforces the idea that, for many real-world applications, efficient transformer models may offer the best trade-off between accuracy and resource consumption.

In terms of raw performance, SpanBERT-base emerged as the best-performing model across both SQuAD v1.1 and v2.0, confirming expectations based on its architecture. ALBERT-v2 followed closely, showing excellent results despite its reduced parameter count. BERT-base, while slightly behind the other two, still proved to be a solid baseline, maintaining competitive EM and F1 scores that do not fall far from the top-performing models. These results confirm that all three architectures are well-suited for extractive QA, with meaningful trade-offs in terms of size, efficiency, and accuracy.

Overall, ALBERT emerged as the best-time performing model in our setup, especially on SQuAD v1.1. Interestingly, our results surpass those reported in the original literature. This discrepancy is not unexpected, as the reference scores are based on ALBERT v1, while we employed the improved ALBERT v2 variant. ALBERT v2 introduces refinements such as deeper cross-layer parameter sharing and better training dynamics, which likely contributed to the performance boost observed in our experiments.

SpanBERT achieved the best overall results in our experiments, as expected given its architecture specifically designed to improve span-level predictions. Unlike traditional masked language models, SpanBERT is pre-trained to predict entire spans of text rather than individual tokens, making it particularly well-suited for extractive question answering tasks that involve identifying precise answer spans within a passage.

Despite the differences between the models reported in the literature and those tested in this work, the results obtained in our experiments are fully comparable with publicly available benchmarks. In particular, our performance metrics are consistent with those reported by the Hugging Face community for the same model versions, confirming the validity and reproducibility of our evaluation setup.

As anticipated, the results obtained on SQuAD v1.1 were consistently higher than those on SQuAD v2.0 for all models evaluated. This performance gap is expected and well-documented in the literature, due to the increased complexity and ambiguity introduced in version 2.0 of the dataset.

Unlike SQuAD v1.1, where every question has a guaranteed answer span within the context, SQuAD v2.0 includes unanswerable questions, which require models to not only identify relevant spans but also to learn when no answer is present. This additional layer of difficulty naturally lowers the EM and F1 scores across the board, as models must strike a balance between extracting correct answers and avoiding false positives.

Our experiments reflected this trend clearly: for each model, both EM and F1 scores were higher on SQuAD v1.1 compared to SQuAD v2.0, highlighting how the nature of the task influences model performance.

Closely related to this observation, we also note a significant contrast when comparing model predictions to human performance benchmarks. On SQuAD v1.1, all tested models outperform the average human annotator, as measured by Exact Match and F1 scores. This is a well-known phenomenon in extractive QA tasks over SQuAD v1.1, where models, especially transformer-based ones, can become highly effective at identifying the most likely answer span within well-formed contexts.

However, on SQuAD v2.0, the situation reverses: none of the tested models surpass human-level performance, which remains consistently higher in both EM and F1. This discrepancy highlights a critical limitation in current QA systems, namely, the difficulty in accurately detecting unanswerable

questions, a task that humans are naturally better at due to their broader reasoning capabilities and pragmatic understanding.

This performance gap underscores how SQuAD v2.0 is a much more realistic and challenging benchmark, requiring models not only to extract information, but also to decide when abstention is the correct answer, a capability that still presents challenges for even the most advanced transformer-based architectures.

Another relevant aspect of our experimental setup concerns the model selection strategy. For each model and dataset, we performed random search over the main hyperparameters — specifically, learning rate, number of epochs, and batch size — within ranges recommended by existing literature and online resources (such as implementation repositories and forum discussions). Typically, we ran 2 to 3 training sessions per configuration, selecting the best-performing model based on validation metrics.

In the case of SpanBERT-base, the situation was slightly different: we did not find explicit recommendations for this specific variant in the literature. As a result, we adopted the hyperparameter ranges suggested for SpanBERT-large, adapting them via random selection to suit our computational budget. While this introduces some uncertainty in the optimization, it also reflects realistic conditions where full grid search is infeasible, and suggests that even without exhaustive tuning, SpanBERT-base can still deliver strong performance.

5.1 Limitations

One of the most significant limitations of this project comes from the computational constraints of the environment in which the experiments were conducted. Specifically, Google Colab provides access to NVIDIA T4 GPUs only for limited sessions and with restricted availability. This limitation, combined with the substantial time required to fine-tune transformer-based models, significantly constrained our ability to perform extensive hyperparameter tuning and to experiment with a broader range of model architectures.

Additionally, the limited memory capacity of the T4 GPU prevented us from employing the larger versions of several models, including SpanBERT-large, even when such variants are known to achieve superior performance in the literature. As a result, we were constrained to using SpanBERT base model variants, which typically contain around 110 million parameters, compared to the approximately 340 million parameters in their large counterparts.

Despite these hardware limitations, our results remain competitive. In particular, while our scores for SpanBERT are lower than the reference values reported in the literature, this discrepancy is largely attributable to our use of SpanBERT-base. Nevertheless, SpanBERT-base consistently outperformed both BERT-base and ALBERT-v2 in our experiments, confirming the architectural advantages of SpanBERT for span-level prediction tasks. This suggests that even with limited resources, meaningful performance gains can still be achieved through careful model selection and tuning.

Lastly, it is important to notice that our results were obtained using a slightly reduced training set, as we reserved 10% of the official training data for validation purposes. This reduction can have some impact on the models' ability to generalize, potentially leading to marginally lower scores compared to those reported in original papers, which typically rely on the full training set. Nevertheless, the consistency of the relative rankings and the closeness of the absolute scores suggest that the models remained robust even under this constraint.

6 Conclusions

In this project, we successfully implemented and evaluated an extractive Question Answering system using transformer-based models on Wikipedia passages. Our approach, centered on fine-tuning BERT, AlBERT, and SpanBERT, confirmed the strong capabilities of these architectures in accurately identifying answer spans within given contexts.

The experimental results demonstrated that AlBERT and SpanBERT generally outperformed BERT, aligning with expectations that more specialized or optimized BERT-based models yield better performance for extractive QA tasks. Notably, SpanBERT showed the best overall performance, though its results were slightly lower than reported in some reference papers due to our use of the

"base" version instead of the "large" version. The performance on SQuAD 1.1 was, as expected, better than on SQuAD 2.0, highlighting the increased difficulty introduced by unanswerable questions. Our optimized hyperparameter configurations, determined through systematic random search on a validation set, underscoring the importance of this model selection phase.

The project also highlighted practical considerations, such as memory constraints influencing model selection (e.g., choosing "bert-base" over "bert-large") and the necessity of custom postprocessing for SQuAD 2.0 to correctly handle unanswerable questions.

Future work could explore several developments. Firstly, investigating larger model variants like SpanBERT-large or RoBERTa-large, perhaps with access to more substantial computational resources, could further improve performance. Secondly, a deeper analysis of the specific instances in which models struggle with unanswerable questions in the SQuAD 2.0 dataset could contribute to the development of more effective "no-answer" prediction strategies. Finally, expanding the scope to include generative QA approaches or hybrid RAG models could offer greater flexibility and summarization capabilities, while addressing hallucination remains a critical challenge.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1171. URL https://aclanthology.org/P17-1171/.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question answering in context. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1241. URL https://aclanthology.org/D18-1241/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new qa dataset augmented with context from a search engine, 2017. URL https://arxiv.org/abs/1704.05179.

Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Convbert: Improving bert with span-based dynamic convolution. In *NeurIPS*, 2020.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020. doi: 10.1162/tacl_a_00300. URL https://aclanthology.org/2020.tacl-1.5/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav

Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703/.

Jiawei Li and Yue Zhang. The death of feature engineering? bert with linguistic features on squad 2.0, 2024. URL https://arxiv.org/abs/2404.03184.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. COVID-QA: A question answering dataset for COVID-19. In Karin Verspoor, Kevin Bretonnel Cohen, Mark Dredze, Emilio Ferrara, Jonathan May, Robert Munro, Cecile Paris, and Byron Wallace, editors, *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online, July 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.nlpcovid19-acl.18/.

Kate Pearce, Tiffany Zhan, Aneesh Komanduri, and Justin Zhan. A comparative study of transformer-based language models on extractive question answering, 2021. URL https://arxiv.org/abs/2110.03142.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124/.

Santhosh Thottingal. Question-to-question retrieval for hallucination-free knowledge access: An approach for wikipedia and wikidata question answering, 2025. URL https://arxiv.org/abs/2501.11301.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In Phil Blunsom, Antoine Bordes, Kyunghyun Cho, Shay Cohen, Chris Dyer, Edward Grefenstette, Karl Moritz Hermann, Laura Rimell, Jason Weston, and Scott Yih, editors, *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2623. URL https://aclanthology.org/W17-2623/.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1237. URL https://aclanthology.org/D15-1237/.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL `https://aclanthology.org/D18-1259/`.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

A Appendix / supplemental material

This section provides additional information to support the experiments presented in the main body of this work. In particular, we report:

- the key hyperparameters explored during the model selection phase;
- the corresponding final settings used for each model trained on SQuAD v1.1 and SQuAD v2.0;
- the total training time required for fine-tuning each model.

The goal of the hyperparameter analysis was to identify the most effective configurations in terms of learning rate, batch size, number of training epochs, weight decay, and other relevant factors that influence the performance of transformer-based models in extractive question answering tasks.

The following tables summarize:

- the range of hyperparameter values tested for each model (BERT, ALBERT, SpanBERT);
- the final selected configuration, highlighted in bold, based on the highest validation F1 score;
- the total training time (in minutes) required to complete all epochs on the SQuAD v1.1 and SQuAD v2.0 datasets.

These results offer a comprehensive overview of both the tuning effort and the computational cost associated with fine-tuning each model.

Table 3: Hyperparameters tested during the model selection phase on SQuAD v1.1. The best configuration is highlighted in bold.

Model	τ	epochs	batch	weight decay
BERT	$\{$**5e-5**, $3e-5\}$	$\{$**3**$\}$	$\{48,$ **32**$\}$	$\{0.05,$ **0.01**$\}$
AlBERT	$\{5e-5,$ **3e-5**$\}$	$\{$**2**$\}$	$\{$**32**$, 16\}$	$\{$**0.05**$, 0.01\}$
SpanBERT	$\{$**5e-5**$, 2e-5\}$	$\{$**4**$\}$	$\{$**32**$\}$	$\{0.03,$ **0.01**$\}$

Table 4: Hyperparameters tested during the model selection phase on SQuAD v2.0. The best configuration is highlighted in bold.

Model	τ	epochs	batch	weight decay
BERT	$\{$**5e-5**, $3e-5\}$	$\{$**3**$, 4\}$	$\{48,$ **32**$\}$	$\{0.05,$ **0.01**$\}$
AlBERT	$\{5e-5,$ **3e-5**$\}$	$\{$**3**$\}$	$\{$**16**$\}$	$\{$**0.01**$\}$
SpanBERT	$\{$**5e-5**$, 2e-5\}$	$\{$**4**$\}$	$\{$**32**$\}$	$\{$**0.01**$\}$

Table 5: Total fine-tuning time (in minutes) required for all epochs of each model on SQuAD v1.1.

Model	Training time
BERT	71m
AlBERT	63m
SpanBERT	133m

Table 6: Total fine-tuning time (in minutes) required for all epochs of each model on SQuAD v2.0.

Model	Training time
BERT	110m
AlBERT	140m
SpanBERT	205m