BERT, Llama and Augmentations for Extractive Question Answering on SQuAD v1.1

Kim Young Jin kim.young.jin@u.nus.edu
Lai Foong Ming foongming@u.nus.edu
Kumaraguru S/O Sreenivasan e1122462@u.nus.edu
Sia Zhi Hong zhihong@u.nus.edu
Yim Sohyun e0960421@u.nus.edu

Abstract

This work presents a comparative study of encoder-based and decoder-based transformer architectures for extractive Question Answering (QA) on the SQuAD v1.1 dataset. Using BERT and LLaMA as representative models, we evaluate the impact of three helper methods: parameter-efficient fine-tuning via Low-Rank Adaptation (LoRA), dependency parsing, and synthetic data generation. Across four experimental settings for each architecture, we conduct controlled ablation studies and hyperparameter tuning to isolate the contribution of each technique. Our results indicate that while the BERT baseline outperforms the LLaMA baseline on this task, the effectiveness of the augmentations varies substantially between architectures. Based on these findings, we construct a final QA system by integrating BERT fine-tuning with Dependency Parsing and Synthetic Data generation, achieving a final EM of 80.9 and F1 of 88.4.

1 Introduction

Question answering (QA) is a key benchmark in natural language processing (NLP) that tests a model's ability to comprehend and extract information from text. The Stanford Question Answering Dataset (SQuAD v1.1) (Rajpurkar et al., 2016) is one such example. An extractive QA model is provided a large segment of text, called the context, and a question, often a sentence long. The task is to predict the exact span, or contiguous substring, of the context that answers the question.

This work aims to train a model to perform on SQuAD v1.1. To do so, we explore BERT as a representative for encoders (Devlin et al., 2018) and LLaMA as a representative for decoders(Zhang et al., 2024; Dubey et al., 2024), as well as a few augmentations: (1) LoRA (Hu et al., 2021) for parameter-efficient adaptation, (2) incorporating dependency parsing features to introduce syntactic information, and (3) adding synthetic question–answer pairs to expand the training data. We then combine the best augmentations to form our final model.

2 Related Work

Low-Rank Adaptation (LoRA) has been shown to improve large language model performance across various benchmark tasks (Hu et al., 2021), with stronger gains observed in decoder-based architectures than in encoder-based ones.

Synthetic data generation using LLMs has been shown to improve performance when used to augment datasets especially where human-labeled data is limited (Nadăș et al., 2025), such as in SQuAD v1.1.

Previous work has also shown that pre-trained models like BERT can benefit from syntactic relationship signals in question answering (Xu et al., 2022). Dependency Parsing can provide explicit syntactic cues to augment embeddings of models like BERT and LLaMA. This could help the model better understand relationships between question and context tokens, improving its ability to locate the correct answer span.

3 Methodology

We rely on Huggingface's Transformers library ((Wolf et al., 2019)) as a standard interface to interact with models.

3.1 Overview of Models

Table 1 provides an overview of the models and helper methods used in this study. We conducted an ablation study for each architecture type, systematically introducing the augmentations to assess their individual contributions. The final hyperparameters used for all models are listed in Table 2.

For our experimental setup, we processed the official SQuAD v1.1 dataset. The original training set, which contained 87599 question-answer pairs, was partitioned into a new training set and a validation set using a 90/10 split. This resulted in 78839 samples for training and 8760 samples for validation. The original SQuAD v1.1 development set, which includes 10,570 samples, was held out and used as the final test set. All splits were performed deterministically using a fixed random seed to ensure reproducibility. The samples were subsequently converted into Hugging Face Dataset objects to standardize data loading and preprocessing across experiments.

Where hyper parameters are tuned, to increase efficiency, the grid search was only performed on a random subset of data. The model is then fine tuned using the full set of data using the parameters that provide the highest EM and F1 scores.

3.2 BERT

We first implement `google-bert/bert-base-uncased` (Devlin et al., 2018) with two classification heads to serve as the baseline model. After hyperparameter tuning and full training, our fine tuned model achieved a comparable EM and F1 of 79.9 and 87.55, against the original work (80.8 and 88.5 respectively).

3.3 Llama

To compare decoder-based and encoder-based architectures under identical experimental conditions, we implemented Llama from `TinyLlama/TinyLlama-1.1B-Chat-v1.0` with two span-prediction heads similar to those used in the BERT baseline. The model architecture extends the causal language modeling backbone by attaching two linear classification heads for predicting the start and end positions of the answer span within the input context. Hyper parameter tuning followed the same grid search procedure as for BERT.

3.4 LoRA

LoRA was implemented on top of the base BERT and Llama models. Base model weights were kept frozen while additional parameters were added to each model. We use the PEFT library (Mangrulkar et al., 2022) for the LoRA implementation.

For BERT, grid search for hyperparameters was done on 108 combinations. LoRA was applied to the self-attention layers (query, key, value) as well as the dense layers in both the attention and feed-forward (MLP) blocks, while excluding the classification heads.

For Llama, we faced significant computational limitations which greatly influenced our implementation. With our limited GPU memory, we could not use LoRA on TinyLlama-1.1B in full precision, and therefore had to quantize to 4-bit with the `BitsAndBytes` library by Dettmers et al. (2023). Considering the influence of quantization, we decided to upgrade to `meta-llamaLlama-3.1-8B` (Dubey et al., 2024), as in our testing this had comparable GPU memory requirements as TinyLlama-1.1B, despite the difference in number of parameters. Another limitation is the hyperparameter search, where manual search was attempted to find an adequately-performing set of hyperparameters. Similar to the BERT implementation, LoRA was applied to all self-attention layers as well as dense layers, excluding the output head.

3.5 Dependency Parsing

SpaCy was used to generate the syntactic dependency labels required for our dependency-path (DP) features. Specifically, different sizes of SpaCy's English models (`en_core_web_sm`, `en_core_web_md`, and `en_core_web_lg`) were tested to extract dependency relations such as subject, object, and modifier links from the context passages.

For BERT, we encoded each token's path to its syntactic governors as a fixed-length sequence and processed it via a small GRU encoder before fusing it with the contextual embeddings. This approach is motivated by prior work showing that syntactic guidance through dependency paths can improve reading comprehension in encoder-based models (Zhang et al., 2020; Xu et al., 2022).

For Llama, we extended this idea by also including part-of-speech and relative-head-distance features, embedding each type and combining them with the token embeddings using either simple concatenation or a small MLP. Incorporating these additional structured features is inspired by methods that inject linguistic knowledge into pre-trained models to enhance their reasoning capabilities (Wang et al., 2020), and applying it to decoder-based models like LLaMA remains largely unexplored, making this a novel approach for improving question answering performance.

3.6 Synthetic Data

To generate synthetic data, we mimic SQuAD's original process as faithfully as possible, while replacing human labour with LLM generation.

SQuAD passages were sourced from the top 10,000 English Wikipedia articles as ranked by Project Nayuki's internal PageRanks (Nayuki, 2016). The original dataset randomly sampled 536 articles from this set and we sample an additional 100 articles. Following Rajpurkar et al. (2016), we extract individual paragraphs and filter those containing fewer than 500 characters. While the original dataset employed crowdworkers to create question-answer pairs, we use an open-source LLM to generate synthetic pairs for our sampled articles.

We generate question-answer pairs using a few-shot prompting approach. The prompt includes example QA pairs randomly sampled from the SQuAD training set, the target paragraphs from our 100 articles, and formatting guidelines specifying JSON output. The LLM is instructed to follow the SQuAD data structure.

We observe that LLMs reliably extract answer substrings but struggle with accurate character indexing. We therefore apply a post-processing script to verify and correct all `answer_start` indices by matching the answer text to its actual position in the context.

We selected the `Kimi-K2-Instruct-0905` open-weight LLM (Moonshot AI, 2025) for synthetic data generation based on a manual evaluation of several leading open-weight models, including `gpt-oss-20b` (Agarwal et al., 2025) and `Qwen3-Next-80B-A3B-Instruct` (Qwen Team, 2025). While output quality was comparable across models, Kimi offered superior cost-efficiency and a larger context window.

4 Results

Table 2 presents performance on the SQuAD v1.1 test set. On BERT, dependency parsing yielded the largest gains (EM: 80.8, F1: 88.3), with the combined DP+SD model achieving best performance (EM: 80.9, F1: 88.4). For LLaMA, LoRA produced the most substantial improvements (EM: 80.1, F1: 88.7).

5 Discussion

5.1 LoRA

Contrary to our expectations, BERT did not benefit significantly from LoRA adaptations. This aligns with the original observations by Hu et al. (2021) that encoder-only models like RoBERTa showed mixed or negative results with LoRA on some benchmark GLUE tasks, suggesting LoRA may be better suited for decoder architectures.

Accordingly, we see significant performance gains when using LoRA with Llama. While it is unclear how much of this is due to a superior base model (Llama 3.1 8B vs TinyLlama 1.1B), we find empirically that LoRA with quantization enables the 8B model to be a viable option given our resource constraints. In comparison, the 1.1B model performed poorer, even at full precision.

5.2 Dependency Parsing

As hypothesized, integrating DP features into the pre-trained models BERT and Llama managed to improve their performance. We trained a total of 9 BERT variants and 36 LLaMA variants with different dependency parsing features, SpaCy model sizes, and fusion strategies.

For BERT, it is noticed that increasing the size of the SpaCy model used for the parser in general showed small improvements in the performance. The best model uses the `en_core_web_lg parser` and does not include padding to the maximum context length of the model.

For LLaMA, the best-performing models also utilised the larger SpaCy parsers. However, this wasn't consistent across the different variations tested with performance dipping in some cases. Notably, models incorporating multiple features (DEP+POS) showed an improvement over those that contain a single type only (DEP), suggesting that richer syntactic information can help but may require careful tuning. Overall, the results indicate that even small dependency embeddings fused with the base model can provide measurable gains, and model size and padding strategies influence stability and performance.

5.3 Synthetic Data

We suspect that increasing the size of the training set using synthetic data caused greater improvement for Llama compared to BERT due to scaling laws (Kaplan et al., 2020). Llama has orders of magnitude more parameters than BERT. We hypothesize that BERT had already reached close to its saturation point on the original SQuAD train set, while Llama had much more remaining capacity it could use to exploit an increased amount of data.

Model	Configuration	Description
BERT (Encoder)	Standard QA	Fine-tuning with dual QA heads for extractive span prediction.
BERT (Encoder)	+ LoRA	Parameter-efficient fine-tuning using Low-Rank Adaptation.
BERT (Encoder)	+ Dependency Parsing	Incorporating word label information from dependency parsing
BERT (Encoder)	+ Synthetic Data	Training with additional generated question–answer pairs.
LLaMA (Decoder)	Standard QA	Fine-tuning with dual QA heads for extractive span prediction.
LLaMA (Decoder)	+ LoRA	Parameter-efficient fine-tuning using Low-Rank Adaptation.
LLaMA (Decoder)	+ Dependency Parsing	Incorporating word label information from dependency parsing.
LLaMA (Decoder)	+ Synthetic Data	Training with additional generated question–answer pairs.
BERT (Encoder)	+ DP, Syn	Incorporating dependency parsing and augmented training data.
LLaMA (Decoder)	+ LoRA, Syn	Incorporating Low-Rank Adaptation and augmented training data.

Table 1: Overview of experimental setups for encoder-based (BERT) and decoder-based (LLaMA) QA models on SQuAD v1.1. Each configuration is tuned for optimal hyperparameters and evaluated through ablation studies.

Model	Configuration	EM	F1	Batch Size	LR	Epochs	Max Length	Doc Stride	LoRA r	LoRA Alpha	LoRA Dropout
BERT	Standard QA	79.9	87.6	16	3e-5	2	384	128	–	–	–
BERT	+ LoRA	80.1	87.2	16	5e-4	3	384	128	32	64	0.1
BERT	+ Dependency	**80.8**	**88.3**	16	3e-5	2	512	384	–	–	–
BERT	+ Synthetic	80.3	87.9	16	3e-5	2	384	128	–	–	–
LLaMA	Standard QA	75.6	83.9	4	3e-5	2	512	128	–	–	–
LLaMA	+ LoRA	**80.1**	**88.7**	16	2e-4	2	384	128	8	16	0.2
LLaMA	+ Dependency	76.0	84.6	4	3e-5	2	512	128	–	–	–
LLaMA	+ Synthetic	77.9	86.1	4	3e-5	2	512	128	–	–	–
BERT	+ DP, Syn	**80.9**	**88.4**	16	3e-5	2	512	384	–	–	–

Table 2: Hyperparameters and performance metrics for encoder-based (BERT) and decoder-based (LLaMA) QA models on SQuAD v1.1. LoRA parameters are only applicable to configurations using LoRA. The best EM and F1 in each category is bolded.

5.4 Combined Model

Given BERT's superior training efficiency and overall performance relative to LLaMA, we combined both DP and SD augmentations for BERT. Although this compound approach yielded our best-performing model, the incremental improvement was marginal, indicating diminishing returns when stacking multiple augmentation strategies.

6 Limitations

This study employed LoRA for parameter-efficient fine-tuning due to computational constraints. With additional resources, future work could explore full fine-tuning, larger model architectures, increased synthetic data generation, or alternative approaches including reasoning models, few-shot prompting, and ensemble methods. Our exploration of augmentation combinations was necessarily limited. While our results suggest that combining Llama 8B with quantization, LoRA, synthetic data, and dependency parsing would be promising, systematic investigation of this and other augmentation strategies remains for future work. Additionally, we generated synthetic data from only 100 additional articles - investigating the effects of scaling to 200, 300, or more articles, and whether performance follows established scaling laws, presents an interesting direction for future research.

7 Conclusion

We identify various augmentations and compare their individual effects in an ablation study. After identifying the most impactful augmentations across models, we combine them to form our final model.

For our submission, we select BERT fine-tuning with Dependency Parsing and Synthetic Data generation based on its computational efficiency and empirically demonstrated performance. Future work could extend to multilingual QA benchmarks, incorporate inference speed metrics, optimise for edge device deployment, or incorporate unanswerable questions such as in SQuAD 2.0 (Rajpurkar et al., 2018).

References

Sandhini Agarwal et al. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv*.

Tim Dettmers, Artidoro Pagnoni, et al. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Jacob Devlin, Ming-Wei Chang, et al. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The llama 3 herd of models.

Edward J. Hu, Yelong Shen, et al. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Jared Kaplan, Sam McCandlish, et al. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Sourab Mangrulkar, Sylvain Gugger, et al. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Moonshot AI. 2025. Kimi-k2-instruct-0905: Model card. `https://huggingface.co/moonshotai/Kimi-K2-Instruct-0905`.

Mihai Nadăș, Laura Dioșan, et al. 2025. Synthetic data generation using large language models: Advances in text and code. *IEEE Access*.

Project Nayuki. 2016. Computing wikipedia's internal pageranks.

Qwen Team. 2025. Qwen3-next-80b-a3b-instruct: Model card. `https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct`.

Pranav Rajpurkar, Robin Jia, et al. 2018. Know what you don't know: Unanswerable questions for squad. *ArXiv*, abs/1806.03822.

Pranav Rajpurkar, Jian Zhang, et al. 2016. Squad: 100,000+ questions for machine comprehension of text. *Preprint*, arXiv:1606.05250.

Ruize Wang, Duyu Tang, et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Proceedings of AAAI*.

Thomas Wolf, Lysandre Debut, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Chuanyun Xu, Zixu Liu, et al. 2022. Multigranularity syntax guidance with graph structure for machine reading comprehension. *Applied Sciences*, 12(19):9525.

Peiyuan Zhang, Guangtao Zeng, et al. 2024. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

Zihan Zhang, Jiawei Yan, et al. 2020. Sg-net: Syntax-guided machine reading comprehension. *AAAI*, 34(05):9982–9990.