BERT-Based Chatbot for Question Answering : A Deep Learning Approach

Damilola Eniola Olowu (DistilBERT), Daeul Lee (BERT), Quynh Nguyen (ALBERT)

December 2nd 2024

1 Introduction

The objective of this project is to develop a chatbot capable of performing question answering using natural language processing techniques, specifically leveraging three transformer-based algorithms. The chatbot is designed to interpret user queries and provide accurate, relevant responses based on the input. Three variants of Bidirectional Encoder Representations from Transformers (BERT) were explored for this project. First, A Lite BERT (ALBERT), designed for greater efficiency in terms of memory and computational cost, was analyzed. Secondly, Distilled BERT (DistilBERT), a lightweight version of BERT optimized for faster inference, was investigated. Lastly, the standard BERT model was employed as a robust baseline for comparison. Each model offers unique strengths, contributing to the enhancement of the chatbot's overall performance.

From online searches to information retrieval, question answering systems are becoming increasingly popular and widely integrated into our daily lives. In 2016, Rajpurkar et al. [1] introduced the Stanford Question Answering Dataset (SQuAD 1.0), featuring 100,000 question-answer pairs, each associated with a context paragraph. Building on this foundation, Rajpurkar et al. [2] released SQuAD 2.0 in 2018, incorporating over 50,000 unanswerable questions, thereby presenting a significantly greater challenge for model development. With the rise of Artificial Intelligence in contemporary society, it is to no surprise that researchers are testing the boundaries and limits of the knowledge within the field of Natural Language Processing (NLP). Although scientists proceed with caution yet excitement, the field of NLP is set to revolutionize the world.

2 Dataset

The renowned SQuAD 2.0 dataset, a comprehensive resource containing 142,192 examples, is used for this project [2] [3]. This dataset includes a mix of answerable and unanswerable questions, enabling an evaluation of the model's ability to understand and reason about context effectively. Each instance in SQuAD 2.0 comprises a context (a passage of text from Wikipedia articles), a question, and an answer. The answer, when present, is a span of text extracted directly from the context; otherwise, the answer is marked as unanswerable.

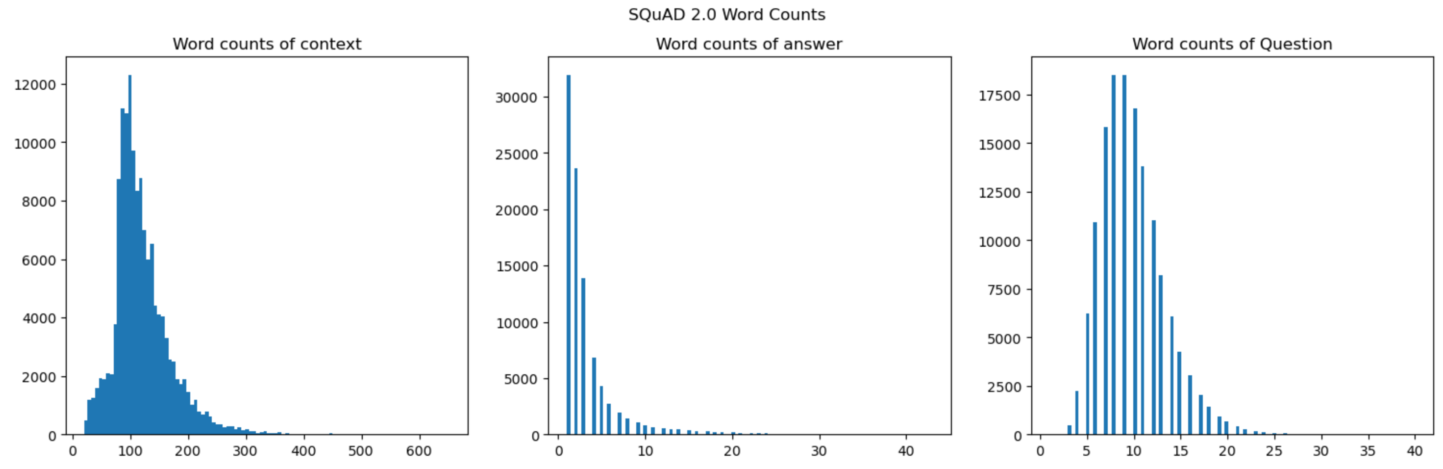


Figure 1: SQuAD 2.0 Word Counts

The data is partitioned into 64% for training, 16% for validation, and 20% for testing, ensuring a balanced approach to model development and evaluation. The numerical breakdown of answerable and unanswerable questions is depicted in Figure 2.

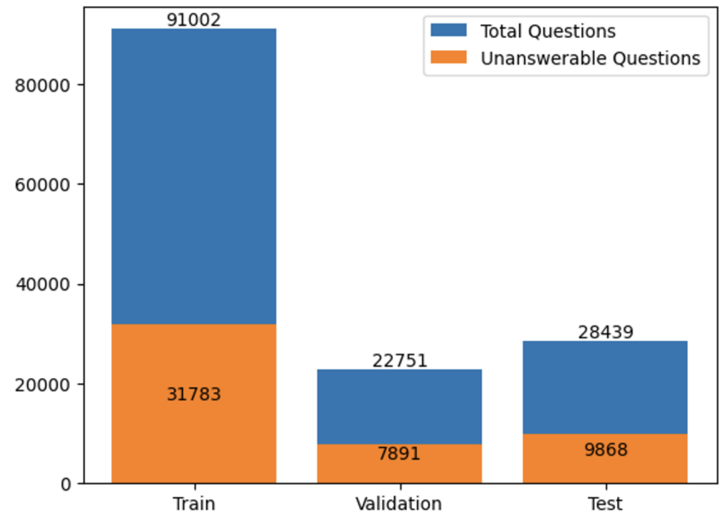


Figure 2: Answerable vs Unanswerable Questions before Preprocessing

3 Methodology

In this project, all models (ALBERT, DistilBERT, and BERT) were evaluated using the following metrics to assess their performance on the SQuAD 2.0 dataset for question answering (QA) task:

1. **Evaluation Metrics**

   (a) Exact Match: The percentage of prediction that match any of the ground truth answers exactly. If the predicted answer matches the true answer character-for-character, it scores 1; other wise, it scores 0. This is a strict metric that reflects how accurately a model can reproduce the exact answer from the text.

$$\text{EM} = \frac{\text{Number of Exact Matches}}{\text{Total Number of Samples}} \times 100 \tag{1}$$

   (b) Precision: measures the fraction of correctly predicted tokens among the total predicted tokens

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

   (c) Recall: measures the fraction of correctly predicted tokens among the ground truth tokens

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

   (d) F1 Score: The average F1 scores evaluates the overlap between the predicted tokens and the ground truth labels. This balances precision and recall, providing a more nuanced assessment of model performance.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

   Both exact match and F1 Score ware used to compare different model's performances on SQuAD 2.0 dastaset. [2]

2. **ALBERT**

(a) **Model Architecture**

ALBERT shares the same Transformer-based backbone as BERT, with both models built on the attention mechanisms fundamental to the Transformer architecture (illustrated in Figure 3 which was obtained from [4]). During the pre-training phase, both models tokenize input text into smaller units, adding special tokens like [CLS] to indicate the start and [SEP] to separate text segments. Both utilize Masked Language Modeling (MLM) to predict masked tokens in a sequence. However, ALBERT replaces BERT's Next Sentence Prediction (NSP) task with Sentence Order Prediction (SOP), which better models coherence between sentences. For fine-tuning tasks like QA, ALBERT predicts the start and end positions of answers given a question and context paragraph.

While maintaining the core architecture, ALBERT introduces innovations to reduce model size and training costs without sacrificing performance:

i. **Factorized Embedding Parameterization:** reduces the number of parameters in the embedding layers by setting the vocabulary embedding size much smaller than the hidden size, $E \ll H$, improving parameter efficiency whereas in BERT model, the vocabulary embedding is typically set to match the hidden size, $E \equiv H$.

ii. **Cross-Layer Parameter Sharing:** shares parameters across all Transformer layers, significantly lowering model size.

iii. **ALBERT's Sentence Order Prediction (SOP) :** replaces BERT's NSP, focusing on inter-sentence coherence rather than topic prediction.

These changes make ALBERT much more efficient than BERT. For example, BERT-base has 110M parameters, whereas ALBERT-base-v2 has just 11M. Further details of the model configurations and experimental setups are summarized in Table 1.

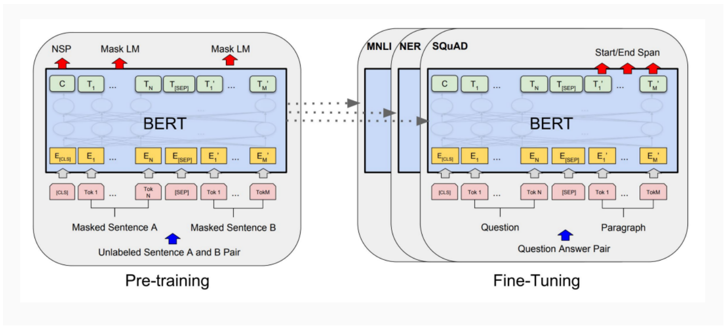


Figure 3: BERT and ALBERT Model Architecture

(b) **Model Setup**

For this study, ALBERT-base-v2 was used with a configuration of: 12 encoder layers (L), 768 hidden size (H), 128 embedding size (E), 12 attention heads, a 3072 feed-forward layer size, and a vocabulary size of 30,000 tokens [5]. The feed-forward layer size was set to 4H, with the number of attention heads defined as H/64, following the design principles established by Z. Lan et al. [6]. The smaller parameter size of ALBERT allows for computational efficiency while maintaining competitive performance.

(c) **Experiment Setup**

The training and evaluation of the ALBERT models adhered to a consistent experimental setup. We used a batch size of 16 across all configurations and the AdamW optimizer with a learning rate of 3e-5, except for ALBERT 2, where the learning rate was set to 2e-5. Most models were trained for 15,603 steps, while ALBERT 3 was trained for an extended 52,010 steps. Additionally, a learning rate scheduler with linear decay and warmup steps was applied to ALBERT 2 and ALBERT 4. For ALBERT 4, transformer weights were frozen, leaving only the task-specific QA layer unfrozen for fine-tuning.

(d) **Training and Testing Strategy**

The training strategy involved several key steps. Inputs were tokenized using ALBERT-base-v2 (AlbertTokenizerFast), invalid spans for unanswerable questions were marked, and the data was split into 64%

Parameter	**ALBERT 1**	**ALBERT 2**	**ALBERT 3**	**ALBERT 4**
Maximum Sequence Length	512	384	512	512
Total Parameters	11,094,530	11,094,530	11,094,530	11,094,530
Trainable Parameters	11,094,530	11,094,530	11,094,530	1,538
Frozen Parameters	0	0	0	11,092,992
Batch Size	16	16	16	16
Learning Rate	3.00E-05	2.00E-05	3.00E-05	3.00E-05
Epochs	3	3	10	3
Number of Training Steps	15,603	15,603	52,010	15,603
Warmup Steps (10% of total training steps)	1,560	1,560	5,201	1,560
Optimizer	AdamW	AdamW	AdamW	AdamW
Scheduler	Disabled	Linear with Warmup	Disabled	Linear with Warmup
Gradient Clipping	Disabled	Enable	Disabled	Disabled
Gradient Accumulation Step	Disabled	2	Disabled	Disabled
FP16 Training	Disabled	Enable	Disabled	Disabled

Table 1: Hyperparameter and Configuration Setup Across ALBERT Models used in this experiment

training, 16% validation, and 20% testing subsets. The AlbertForQuestionAnswering model, initialized with pretrained ALBERT-base-v2 weights, included task-specific heads for span prediction. Training and evaluation functions computed metrics like Precision, Recall, F1, and Exact Match (EM) scores. Models were fine-tuned for 3 epochs, except ALBERT 3, which was trained for 10 epochs. Evaluation included validation and test sets, with epoch durations recorded to measure computational efficiency.

3. DistilBERT

(a) Model Architecture

DistilBERT, is a distilled version of BERT: smaller, faster, cheaper and lighter [7]. Specifically, it reduces the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster. DistilBERT generally has the same general architecture as BERT (Figure 4). The architecture of DistilBERT is based on the Transformer model, consisting solely of the encoder portion of the original Transformer. DistilBERT includes an embedding layer to process tokenized input, followed by 6 Transformer layers, compared to the 12 Transformer layers in the original BERT model. These Transformer layers use multi-headed self-attention to capture contextual relationships between tokens. Finally, a prediction layer outputs the most likely result. For QA tasks, an additional QA-specific output layer is added on top of DistilBERT's architecture during fine-tuning to enable the model to predict answers from a given context.

(b) Model Setup

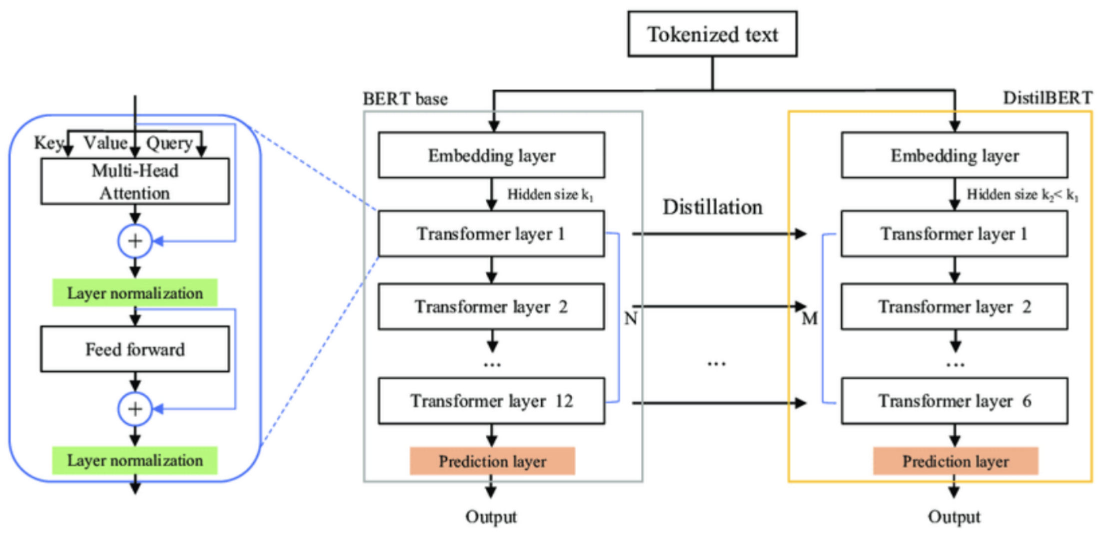
For this study, the DistilBERT-base-uncased model was used with the following configuration: 6 encoder layers (`n_layers`), a hidden size of 768 (`dim`), an embedding size of 768 (`E`), 12 attention heads (`n_heads`), a feed-forward layer size of 3072 (`hidden_dim`), and a vocabulary size of 30,522 tokens (`vocab_size`). This configuration reflects the efficiency of DistilBERT, which reduces the number of encoder layers compared to the original BERT model while maintaining a high level of performance.

(c) Experiment Setup

The training and evaluation of the DistilBERT models followed the outlined experimental setup. A batch size of 16 was maintained consistently across all configurations for both training and evaluation. The AdamW optimizer was used with a learning rate of 2e-5 and a total of 3 epochs for training, with all layers frozen except for the QA head, which had 1,538 trainable parameters. This configuration yielded suboptimal results, as discussed in the results section.

To improve performance, an alternative setup was tested. This involved increasing the number of trainable parameters to 21,263,616 by unfreezing layers 3, 4, and 5 of the DistilBERT model and using a learning rate of 1e-5 with 5 epochs of training. Layers 3, 4, and 5 were specifically chosen because they are closer to the model's output and are more task-specific compared to lower layers, which primarily

Figure 4: BERT and DistilBERT Model Architecture.



capture general linguistic features. By fine-tuning these higher layers, the model can better adapt to the nuances of the question-answering task. This approach resulted in improved accuracy.

Additionally, weight decay was set to 0.01 to regularize the model and mitigate overfitting. The evaluation strategy was configured to execute at the end of each epoch, and model checkpoints were saved after every epoch to preserve intermediate results.

To further optimize the training process, mixed-precision training (fp16) was enabled, providing faster computations and reduced memory usage on compatible hardware.

(d) **Training and Testing Strategy**

The training process involved distinct setups for frozen and unfrozen configurations to evaluate their performance on the SQuAD v2.0 dataset. Initially, the dataset was imported using Hugging Face's datasets library and split into training, validation, and test sets. The dataset was split into 64%, 16%, and 20% proportions. Both configurations included preprocessing steps to handle invalid spans for unanswerable questions and tokenization using AutoTokenizer. Dynamic padding was applied via DataCollatorWithPadding to optimize memory usage during training and evaluation. For both setups, custom Dataset and DataLoader classes handled batching, shuffling, and padding. Metrics such as Exact Match (EM), F1 score, precision, and recall were defined and applied consistently across frozen and unfrozen configurations. Exact Match (EM) checks if the predicted answer exactly matches the ground truth after normalizing both (removing punctuation, articles, and extra spaces, and converting to lowercase). If they match, EM is 1; otherwise, it is 0. F1 score measures the overlap between the predicted and ground truth tokens, balancing precision and recall. Precision calculates the fraction of correctly predicted tokens out of all predicted tokens, while recall measures the fraction of ground truth tokens correctly identified in the prediction. F1 combines these metrics, providing a balanced score. Training for the frozen configuration updated only the weights of the task-specific head, while the unfrozen configuration updated the weights of layers 3, 4, 5, and the task-specific head. After training, both models were evaluated on validation and test datasets. The results showed performance improvements in the unfrozen configuration due to the increased number of trainable parameters and longer training duration. Metrics were noted to compare the configurations, highlighting the trade-offs between computational efficiency in the frozen setup and accuracy improvements in the unfrozen setup.

4. BERT

(a) Model Architecture

BERT is pre-trained on large amount of text data, including Wikipedia and Google's BooksCorpus, totaling about 3.3 billion words. For this fine-tuning task, 'BERT-base-uncased' [8] was used as a tokenizer and base model, which is a specific variant of the BERT model. The model consists of 12 transformer layers, 768 hidden units, and 12 attention heads. The model processes all text as lowercase, ignoring the case of the input text.

A BERT QA system is designed to read a question paired a paragraph, which are separated by the [SEP] token. BERT's question answering system includes the start-end token classifier, which indicates the location of the answer in the provided passage. Scores for potential answer spans are computed by taking the dot product of the start and end vectors with the output vectors from the context. Once scores are obtained, a SoftMax function is applied to convert these scores into probabilities.

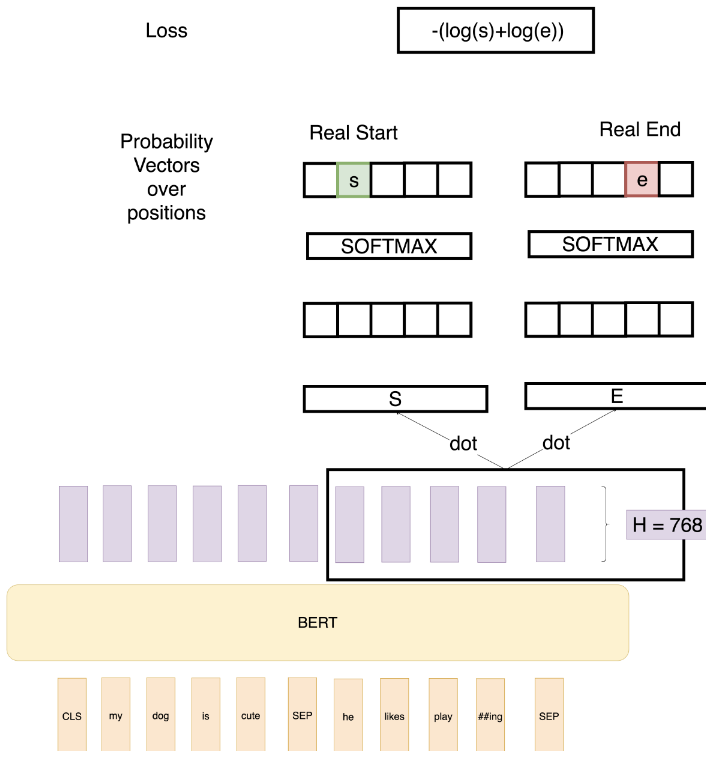


Figure 5: Question Answering System with BERT [9]

(b) **Methodology**

i. Data Preprocessing:

During preprocessing phase, question and context pairs were truncated to maintain a maximum length of 384 tokens, ensuring that all inputs fit within the model's constraints. A stride of 128 was employed to handle cases where tokenized inputs exceeded the maximum length, allowing for the creation of manageable segments. The preprocessing also included padding to ensure uniform input sizes across batch sizes, with batch size set at 16. In addition, answers were assigned to their corresponding token positions and labels were assigned to indicate the start and end positions of the answers, along with labels that indicate whether the answer is present or not.

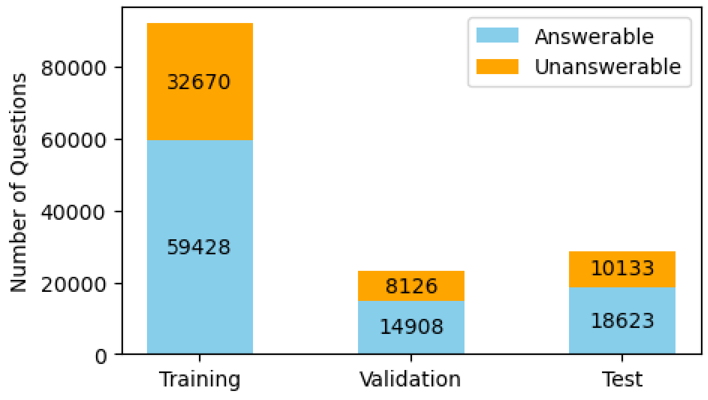


Figure 6: Answerable vs Unanswerable Questions after Preprocessing

ii. During the fine-tuning task, only the start-end token classifier were trained. The no answer probability is calculated based on the length of the answer span fed into the softmax function. This corresponds to the model's increased confidence with longer answers due to richer contextual information supporting prediction. Out of 110 million parameters, only 1,586 parameters were fine-tuned. During hyperparameter tuning, following setting were explored:

Hyperparameter	**Values**
Batch Size	16
Learning Rate	{3e-05, 1e-5, 5e-5}
Number of Epochs	{2, 3, 5}
Adam Optimizer	Optional, weight decay: 0.01, β: (0.9, 0.999), ϵ: 1e-8
Warm Scheduler	Optional, constant, warm-up steps: 500

Table 2: Hyperparameter Settings for BERT

The best hyperparameters was: {Batch size: 16, Learning rate: 5e-5, Number of epochs: 3, with adam optimizer

4 Results

i. **ALBERT**

Metric	**ALBERT 1**	**ALBERT 2**	**ALBERT 3**	**ALBERT 4**
Start Precision	0.4351	0.4273	0.4262	0.0016
Start Recall	0.4351	0.4273	0.4262	0.0016
Start F1	0.4351	0.4273	0.4262	0.0016
Start EM	0.4351	0.4273	0.4262	0.0016
End Precision	0.4778	0.4599	0.4569	0.0066
End Recall	0.4778	0.4599	0.4569	0.0066
End F1	0.4778	0.4599	0.4569	0.0066
End EM	0.4778	0.4599	0.4569	0.0066

Table 3: Performance Metrics Comparison Across ALBERT Models

The alignment of precision, recall, F1, and EM scores is a natural outcome of the question-answering task's design and evaluation framework (Table 3), where the predictions for start and end tokens are evaluated using closely related metrics. The similarity indicates that the models perform consistently across different evaluation criteria. More insights on the result are provided in the Discussion section.

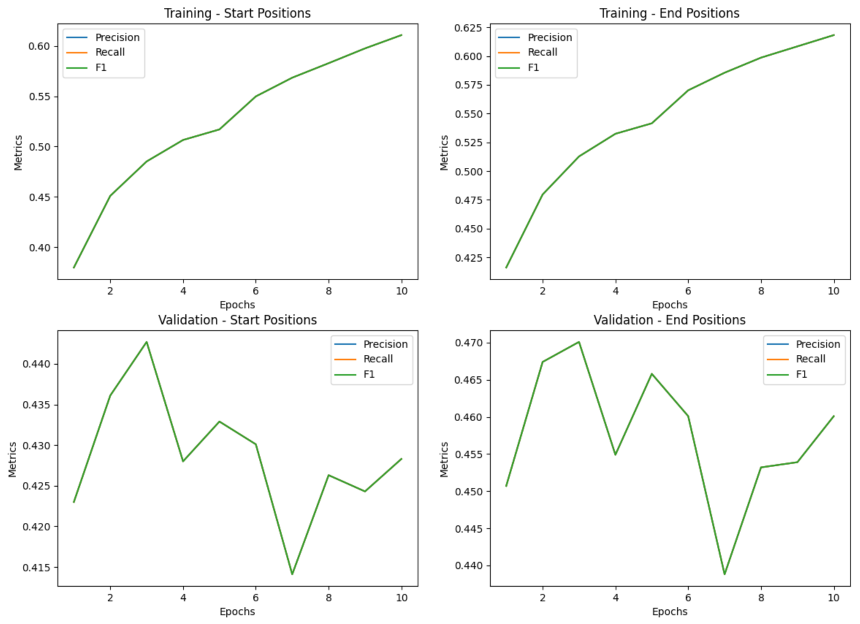


Figure 7: Performance Metrics for ALBERT 3 during Training and Validation

From Figure 7, the performance metrics for ALBERT 3 during training show consistent improvement. By epoch 10, precision, recall, and F1 scores for start positions reach approximately 0.61, while end positions converge around 0.62, reflecting the model's increasing ability to accurately identify the start and end tokens of answer spans. In contrast, validation metrics fluctuate significantly. For start positions, precision, recall, and F1 scores range between 0.42 and 0.44, with no clear upward trend. Similarly, end positions peak early but stabilize around 0.46 to 0.47 by epoch 10. These inconsistencies highlight challenges in generalization to unseen data, likely due to the complexity of the SQuAD 2.0 dataset, which includes both answerable and unanswerable questions.

ii. **DistilBERT**

The training results demonstrate consistent improvements across key evaluation metrics, reflecting the effectiveness of the fine-tuning process as observed in Table 4. The training and validation losses decrease steadily over five epochs, with the training loss reducing from 1.74 to 1.28 and validation loss decreasing from 1.59 to 1.29, suggesting the model is effectively learning the task without significant overfitting. The Exact Match (EM) score, which measures the percentage of predictions that perfectly match the ground truth, improves from 41.37% in epoch 1 to 49.25% in epoch 5. Similarly, the F1 score, which balances precision and recall, rises from 51.13% to 59.98%, indicating the model's increasing ability to capture partial matches and relevant spans. Precision and recall also improve proportionally, reaching nearly 60% by the final epoch.

Epoch	**Training Loss**	**Validation Loss**	**Exact Match (%)**	**F1 (%)**	**Precision (%)**	**Recall (%)**
1	1.7429	1.5947	41.37	51.14	51.14	51.14
2	1.5271	1.4138	45.54	56.47	56.47	56.47
3	1.3562	1.3439	47.40	58.44	58.44	58.44
4	1.2908	1.3119	48.44	59.34	59.34	59.34
5	1.2785	1.2935	49.25	59.98	59.98	59.98

Table 4: Performance Metrics Across 5 Epochs

The training results highlight limited progress across the training process, with key evaluation metrics remaining stagnant as seen in Table 5. The training loss starts at 3.53 and fluctuates slightly, ending at 3.63, while the validation loss decreases marginally from 3.59 to 3.48 over three epochs, suggesting minimal improvement in the model's ability to generalize. The Exact Match (EM) score remains constant at 33.37% across all epochs, indicating that the model is consistently retrieving one-third of the answers correctly without improvement. The F1 score remains at 0.00%, signaling that the model is failing to recognize overlaps or partially correct spans between predictions and ground truths. Similarly, precision and recall remain static at 33.37%, reflecting the lack of growth in accurately identifying and retrieving answer spans.

Epoch	Training Loss	Validation Loss	Exact Match (%)	F1 (%)	Precision (%)	Recall (%)
1	3.5315	3.5970	33.37	0.00	33.37	33.37
2	3.2300	3.5067	33.37	0.00	33.37	33.37
3	3.6313	3.4830	33.37	0.00	33.37	33.37

Table 5: Performance Metrics Across 3 Epochs

iii. **BERT**

A. Performance comparison:

B. Base model: 'bert-case-uncased' [8]

C. Fine-tuned model (QA Layer only, Ours): **Avg. - EM: 32.70, F1: 33.02, HasAns - EM: 0.15, F1: 0.63, NoAns - EM: 95.04, F1: 95.04**

D. Pre-trained model on SQuAD 2.0 without freezing layers [10]

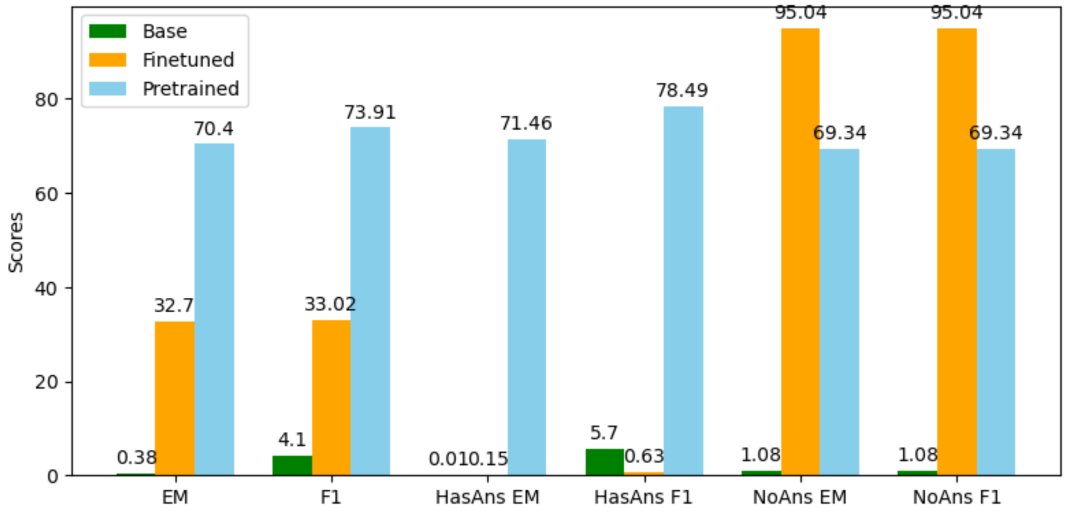


Figure 8: Comparison of Base, Finetuned, and Pretrained Models

To measure the models' performance on answerable questions and unanswerable question separately, the F1 and EM scores for each category are compared. These scores were calculated using the standardized evaluation code provided on the official SQuAD 2.0 dataset website [3]. As observed, the overall EM and F1 scores show improvement over the base model. When compared to the pre-trained model on the SQuAD v2 dataset, the pre-trained model exhibits stable performance for both answerable and unanswerable questions. In contrast, the fine-tuned model, which adjusts only the last layer, achieves high scores on unanswerable questions but performs poorly on answerable questions, as evidenced by significantly lower scores.

The current model is facing the following challenges:

i. The model tends to predict *"no answer"* excessively, even for questions that have valid answers. This behavior is reflected in the high NoAns scores and extremely low HasAns scores.

ii. For questions with valid answers, the model struggles to extract them accurately, as indicated by the very low HasAns exact match and HasAns F1 scores.

5 Discussion

(a) **ALBERT**

ALBERT 1 demonstrated the highest overall performance, achieving start and end F1 scores of 0.4351 and 0.4778, respectively. Its balanced precision and recall reflect its ability to make accurate predictions while capturing most correct spans. ALBERT 2 (achieving start and end F1 scores of 0.4273 and 0.4599, respectively) and ALBERT 3 (achieving start and end F1 scores of 0.4262 and 0.4569, respectively) showed slightly lower performance due to differences in configurations. ALBERT 2 used a lower learning rate (2e-5), FP16 training, and a linear learning rate scheduler, which enhanced computational efficiency but did not lead to significant improvements. ALBERT 2's shorter max sequence length (384) also likely contributed to the slight decline in performance by truncating critical context. For ALBERT 3, it was trained for 52,010 steps (10 epochs), likely suffered from overfitting, reducing its generalization capacity (Figure 7). While gradient accumulation (in ALBERT 2) simulated a larger batch size and gradient clipping stabilized training, these techniques provided only marginal benefits, as ALBERT 1 already achieved high stability without them. ALBERT 4's performance was notably poor, with start and end F1 scores of only 0.0016 and 0.0066, respectively. This drastic decline is attributed to freezing almost all transformer layers, leaving only the QA-specific layer trainable. With just 1,538 trainable parameters, compared to over 11M in the other models, ALBERT 4 lacked the capacity to adapt to the nuances of the question-answering task.

The overall performance across all models was relatively low, highlighting the difficulty of the SQuAD 2.0 dataset, which combines answerable and unanswerable questions. Addressing unanswerable questions requires robust reasoning and contextual understanding, which these configurations struggled to achieve. EM scores, which require exact alignment with the ground truth, were particularly challenging to optimize. In contrast, F1 scores, which balance precision and recall, provided a more forgiving metric that acknowledged partial correctness. These results further proved the existing trade-offs in model configurations, such as freezing layers or adjusting learning rates, and emphasize the challenges inherent in fine-tuning models like ALBERT for complex datasets.

(b) **DistilBERT**

Experiment 1: Fine-Tuned DistilBERT with Partially Frozen Layers – Table 4 Performance Metrics Across 5 Epochs

In this experiment, the top three layers of the DistilBERT model (layers 3–5) were unfrozen for selective fine-tuning, while the lower layers remained frozen to preserve pre-trained knowledge. This approach yielded consistent improvements across key evaluation metrics, demonstrating that targeted fine-tuning effectively enhanced the model's performance. Over five epochs, the observed decrease in both training and validation loss indicated successful learning with minimal overfitting. Although the Exact Match (EM) and F1 scores deviated slightly from the original experiment [7], which reported an EM of 77.7% and an F1 of 85.8%, the results were significantly closer than those shown in Table 4, where only the QA head was fine-tuned. These findings underscore the benefits of increasing the training duration to five epochs and unfreezing layers 3–5, as this strategy produced outcomes more aligned with the original benchmarks compared to the results achieved with three epochs.

The results highlight the importance of selectively unfreezing layers in transformer-based models. The gradual decrease in losses and the significant improvement in metrics indicate that this approach balances generalization and task-specific learning effectively. Future improvements may include unfreezing layers 2–5 to allow for greater task-specific adaptation while closely monitoring for overfitting, in addition to the current approach of unfreezing layers 3–5. Furthermore, adapting the training data by incorporating similar question-answering datasets or synthetic data generated through data augmentation techniques could enhance generalization. Lastly, analyzing error cases in detail—focusing on instances where the model fails to predict correct answers—could help identify recurring patterns and inform refinements to preprocessing steps or the model architecture.

Experiment 2: Fully Frozen DistilBERT – Table 5 Performance Metrics Across 3 Epochs

In this experiment, all layers of the DistilBERT model were frozen, leaving only the task-specific head trainable. Unlike Experiment 1, this approach severely restricted the model's ability to adapt to task-specific nuances, resulting in stagnant performance across all evaluation metrics. The Exact Match (EM) score remained constant at 33.37%, indicating that the model consistently retrieved approximately one-third of the answers correctly without improving its understanding of the task. The F1 score stayed at 0.00%, reflecting the model's inability to recognize partially correct answer spans or overlaps between predictions and ground truth. Similarly, precision and recall metrics remained static at 33.37%, demonstrating no improvement in identifying answer spans or differentiating relevant tokens. This experiment highlights the limitations of relying solely on pre-trained knowledge without task-specific adaptation.

The stark contrast between the results of Experiments 1 and 2 underscores the significance of fine-tuning in adapting pre-trained models to downstream tasks. In Experiment 1, selective fine-tuning of the top three layers (3–5) enabled the model to balance pre-trained linguistic understanding with task-specific learning, leading to substantial improvements in EM and F1 scores, precision, and recall. Conversely, in Experiment 2, the fully frozen model lacked the flexibility to refine its predictions, leading to static and suboptimal performance across all metrics. In Experiment 2, the fully frozen setup limited the model's ability to adapt to these preprocessing strategies, highlighting the need for gradient flow through trainable layers for effective optimization. Furthermore, the ability of Experiment 1's model to handle unanswerable questions and improve prediction accuracy emphasized the critical role of selective fine-tuning in achieving better generalization and task-specific adaptability.

Using adaptive learning rate schedules, like cosine annealing or cyclical learning rates, could enhance optimization for both frozen and unfrozen setups by dynamically adjusting the learning rate throughout training. Additionally, integrating task-specific architectures, such as lightweight adapters or Low-Rank Adaptation (LoRA) modules, could enable effective adaptation to downstream tasks without fully unfreezing the pre-trained layers, maintaining computational efficiency while improving task-specific performance. Source code has been submitted (frozen parameters and pre-changes).

(c) **BERT**

For training, the Trainer class in Hugging Face [11] was used, with cross-entropy loss for Bert Question Answering system. Cross-entropy loss is calculated for each of the start and end position predictions and the total loss is the sum of these two losses. This penalizes the model for deviating from the ground truth distribution.

For unanswerable questions, the ground truth for both the start and end positions is the CLS token (index 0). Predicting the CLS token is a simpler strategy for the model to minimize cross-entropy loss during training, as the CLS token is always present in the sequence and occupies a fixed, predictable position.

In the training and test datasets, approximately 35% of the data consisted of unanswerable questions. This distribution likely influenced the model's tendency to minimize the cross-entropy loss by favoring "no-answer" predictions. This behavior corresponds to the EM and F1 scores observed during evaluation.

Considering that a single BERT model achieves high scores on the SQuAD 1.0 dataset when trained and tested exclusively on answerable questions [4], the mixture of answerable and unanswerable questions in the SQuAD 2.0 dataset presents a significant challenge for the model. To address these challenges, hybrid models that combine BERT with other architectures, such as BiDAF, leverage complementary strengths to improve performance. Encoder-decoder architectures on top of BERT have also been proposed to better handle the complexities of the SQuAD 2.0 dataset [12]. Additionally, ensemble approaches combining multiple models are commonly used to enhance robustness and accuracy, as evidenced by the strategies employed in the SQuAD 2.0 leaderboard [3].

6 Conclusion

To conclude, a thorough evaluation of three transformer - based models - ALBERT, DistilBERT and BERT was experimented with for the development of a chatbot capable of question answering using the SQuAD 2.0 dataset. Each transformer based model showcases unique advantages and limitations, illustrating the nuance and complexity of fine-tuning pre-trained models for tasks requiring contextual understanding. ALBERT demonstrated computational efficiency with reduced parameters but struggled with generalization for unanswerable questions. DistilBERT achieved notable performance improvements with selective layer fine-tuning, highlighting the benefits of balancing pre-trained knowledge with task-specific learning. Meanwhile, BERT, despite its robust architecture, exhibited challenges in handling the mix of answerable and unanswerable questions due to a predisposition toward ”no-answer” predictions. Thus, the results emphasize consideration and trade-offs in computational efficiency, generalization, and task-specific adaptation across models. Future work includes enhancements not limited to the exploration of hybrid architectures, dynamic learning rate schedules, as well as the consideration of ensemble methods. Overall, this project presents a well-done work in providing a strong foundation for advancing the capabilities of transformer-based models in NLP applications.

References

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” 2016. [Online]. Available: https://arxiv.org/abs/1606.05250

[2] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” 2018. [Online]. Available: https://arxiv.org/abs/1806.03822

[3] P. Rajpurkar *et al.*, “Squad explorer,” https://rajpurkar.github.io/SQuAD-explorer/, 2024, [Online; accessed 2-December-2024].

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[5] H. Face, “Albert-base-v2,” https://huggingface.co/albert/albert-base-v2, 2024, [Online; accessed 2-December-2024].

[6] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” 2020. [Online]. Available: https://arxiv.org/abs/1909.11942

[7] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” 2020. [Online]. Available: https://arxiv.org/abs/1910.01108

[8] G. Research, “Bert-base, uncased model,” https://huggingface.co/google-bert/bert-base-uncased, 2024, [Online; accessed 2-December-2024].

[9] E. Corporation, “Understanding bert with hugging face,” https://www.exxactcorp.com/blog/Deep-Learning/understanding-bert-with-hugging-face, 2024, [Online; accessed 2-December-2024].

[10] phiyodr, “Bert-base fine-tuned on squad2,” Hugging Face, 2024, [Online]. Available: https://huggingface.co/phiyodr/bert-base-finetuned-squad2. [Accessed: Dec. 2, 2024].

[11] H. Face, “Transformers trainer documentation,” https://huggingface.co/docs/transformers/main_classes/trainer, 2024, [Online; accessed 2-December-2024].

[12] R. Rejimoan, B. Gnanapriya, and J. Jayasudha, “Enhancing question answering with a multidirectional transformer: Insights from squad 2.0,” *SSRG International Journal of Electronics and Communication Engineering*, vol. 11, no. 4, pp. 133–148, 2024.