Impact of Attention Mechanism on Question Answering in Transformers

Manoj Virinchi Chitta
University of Florida
mchitta@ufl.edu

Tarun Reddy Nimmala
University of Florida
tarunred.nimmala@ufl.edu

Sai Aakash Ekbote
University of Florida
sa.ekbote@ufl.edu

Satya Aakash
University of Florida
s.obellaneni@ufl.edu

Rishika Reddy
University of Florida
r.alugubelli@ufl.edu

Abstract

This research explores the understanding of attention mechanisms in Transformer-based models, specifically BERT, to improve their performance on the question-answering task. We aim to identify the optimal number of attention heads that balance accuracy and computational efficiency. By contrasting the fine-tuned BERT model with varying attention heads against an LSTM model without attention mechanisms.

Our research question delves into how BERT's attention mechanism enhances its ability to accurately interpret and respond to questions across a diverse array of subjects compared to LSTM's sequential data processing, particularly in tasks that require a deep understanding of context and nuanced language interpretation. We assess the impact of attention on handling complex queries when context is provided, specifically in the context of extractive question answering. Our experiments reveal that an increase in the number of attention heads significantly enhances model performance up to a threshold, beyond which the benefits diminish and may even degrade due to overfitting. This study addresses the critical need for efficient model fine-tuning strategies in natural language processing and provides insights into the effective deployment of attention mechanisms in language models for enhanced question-answering.

1 Introduction

Question-answering systems transitioned from early rule-based models to machine learning-based systems, and then to Recurrent Neural Network (RNN) approaches like LSTM and GRU. RNNs, while significant, faced challenges with long-term dependencies and slower sequential processing. This led to the adoption of transformer-based models like BERT and GPT, which enhanced domain-specific QA through fine-tuning on specific domain. Recent advancements in Natural Language Processing (NLP) have significantly enhanced the capability of machines to understand and respond to human language, making NLP crucial in various applications ranging from automated customer service to intelligent personal assistants. One of the most challenging tasks in this field is question answering, which requires a model not only to understand the natural language text but also to provide accurate answers from a given context. An example of its necessity is in the domain of healthcare, where patients frequently seek specific information from vast databases of medical literature. Efficiently and accurately navigating these queries is essential for improving user experience and ensuring that reliable information is readily accessible.

This paper investigates the impact of the attention mechanism and determines the optimal number of attention heads within the Transformer-based BERT model, aimed at optimizing its performance for the question-answering task on the dataset. The dataset presents a unique challenge as it includes questions that are purposely unanswerable based on the given context, thus requiring the model to not only find correct answers but also identify when no valid answer exists. By comparing the results of a fine-tuned BERT model with various configurations of attention heads to those of an LSTM model without attention mechanisms, this study seeks to elucidate the role of attention in processing complex queries and improving answer accuracy. We explore the balance between model complexity and computational efficiency to provide insights into effective strategies for enhancing models in complex NLP tasks.

2 Background

In the field of question answering (QA), advanced neural architectures have gradually replaced rule-based models. Early attempts at quality assurance systems mainly depended on manually created rules and pattern matching, which took a lot of manual labor and domain expertise. These systems struggled with the complexity and variability of natural language, but they were good at responding to questions in specific domains. Following the advent of machine learning, researchers turned to statistical models such as RNNs (Recurrent Neural Networks) and Hidden Markov Models (HMMs). RNNs offered a dynamic framework for handling sequence data and made it easier for users to learn word dependencies in questions and contexts (Jurafsky and Martin, 2008). Despite advancements, these models continued to struggle with long-term dependencies and frequently lost context over longer texts—a critical component for accurate question answering.

The advent of Transformer models marked a turning point in NLP, especially for QA tasks. Introduced by Vaswani et al. in their seminal paper "Attention Is All You Need" (Vaswani et al., 2017), Transformer models, eschewing recurrent layers for self-attention mechanisms, could process entire sequences of words simultaneously, providing a global understanding of the context. This ability to capture complex dependencies across a sequence of words enabled the models to achieve remarkable success on various NLP benchmarks. Subsequently, BERT (Bidirectional Encoder Representations from Transformers) and its derivatives further advanced the field by pre-training on vast corpora of text and fine-tuning specific tasks, including QA, achieving unprecedented levels of performance and truly revolutionizing the approach to automated question-answering systems (Devlin et al., 2019).

The pursuit of improving QA (Quality Assurance) systems extends well beyond academia and has significant real-world implications across multiple sectors. For example, in customer service, automated QA systems can reduce response times and improve the accuracy of support by providing answers to frequently asked questions. In the medical field, QA systems can quickly sort through vast amounts of research to offer healthcare professionals evidence-based answers that can greatly impact life-saving decisions. However, the effectiveness of these systems depends on their ability to not only retrieve information but also comprehend and reason about the context. This challenge is further compounded by the introduction of the SQuAD-v2 dataset, which includes unanswerable questions, requiring models to discern when a question cannot be answered given the text. Achieving this level of understanding was previously unattainable with simpler RNN architectures.

3 Methodology

Our study focuses on extractive question answering, where the goal is to locate the exact answer within a given text, unlike generative question answering that involves creating answers from scratch.We initially wanted to consider - cais/mmlu dataset but upon analysis this dataset was tailored for generative models so we transitioned to Squadv2. Correspondingly,we switched from using BLEU and ROUGE evaluation metrics which were our initial considerations to F1 and Exact Match scores. Since these metrics accurately gauge the model's ability to recognize and extract the appropriate text segments as responses, they are more appropriate for extractive quality assurance. This guarantees that our evaluation is in line with the particular requirements of our research project.

3.1 Dataset Overview - Dataset Link

The SQuAD (Stanford Question Answering Dataset) v2 (Rajpurkar et al., 2018) is a benchmark dataset for machine reading comprehension tasks in natural language processing. Specifically, SQuAD v2 is designed for extractive question answering, where the task is to extract the answer to a given question from a provided passage of text. Dataset is designed to pose a significant challenge for machine comprehension models by including unanswerable questions. It builds upon the first version by adding approximately 50,000 unanswerable questions to the existing 100,000 question-answer pairs from SQuAD v1.1. These questions are created to look similar to answerable ones but do not have a correct answer within the provided passages.

The structure of the SQuAD v2.0 dataset includes

"Title" : The title of the Wikipedia article from which the context is extracted.

"paragraphs" : Each title has one or multiple paragraph entries.

"context" : A paragraph from the Wikipedia article that contains the information necessary to answer the accompanying questions.

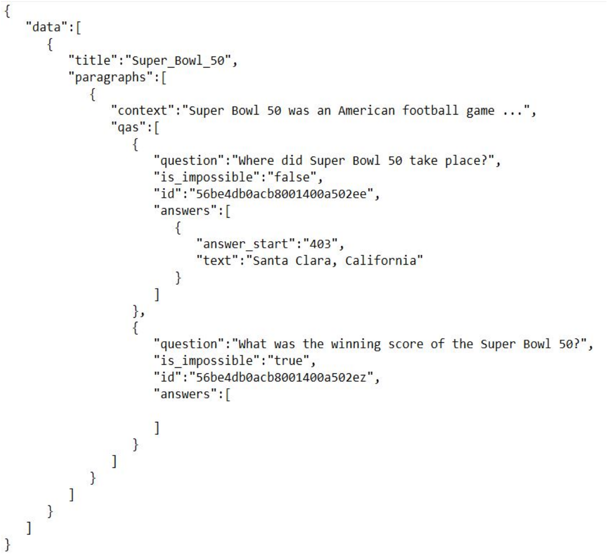
"qas" : Field containing question-answer entries.

Each question-answer entry has the following fields:

- "id" : Unique identifier for each question in the dataset.
- "Context": A paragraph from the Wikipedia article that contains the information necessary to answer the accompanying questions.
- "Question": A question that pertains to the given context. The question may or may not have an answer based on the context provided.
- "Answer": This field contains the answer to a question if one is available. It includes the text of the answer and the indices indicating the position of the answer within the context paragraph.
- "is_impossible": A flag indicating whether a question can be answered with the given context. If true, it means the question does not have an answer in the context.

In the SQuAD v2.0 dataset, each question is structured to elicit a specific answer from the provided context. Figure 1 shows structure of our dataset. If the question is answerable, the dataset contains an answer entry comprising the text span and its corresponding starting character index within the context. However, for questions deemed unanswerable, an empty "answers" list is provided. The questions within SQuAD v2.0 are meticulously crafted to challenge the comprehension and analytical abilities of both humans and machine learning models. These questions encompass various types, including fact-based inquiries seeking precise information from the text, definition queries prompting explanations of terms or concepts mentioned within the context, and reasoning challenges necessitating an understanding of causality, implications, or other nuanced relationships within the provided information. Through this diverse array of question types, SQuAD v2.0 effectively evaluates the capability of models to comprehend and analyze textual content across a range of cognitive tasks.

SQuAD v2.0 thus presents a dual challenge for NLP models: not only must they find answers to direct questions where an answer exists, but they must also identify when no suitable answer is present, testing both the retrieval and comprehension capabilities of the models.



```
{
   "data":[
      {
         "title":"Super_Bowl_50",
         "paragraphs":[
            {
               "context":"Super Bowl 50 was an American football game ...",
               "qas":[
                  {
                     "question":"Where did Super Bowl 50 take place?",
                     "is_impossible":"false",
                     "id":"56be4db0acb8001400a502ee",
                     "answers":[
                        {
                           "answer_start":"403",
                           "text":"Santa Clara, California"
                        }
                     ]
                  },
                  {
                     "question":"What was the winning score of the Super Bowl 50?",
                     "is_impossible":"true",
                     "id":"56be4db0acb8001400a502ez",
                     "answers":[

                     ]
                  }
               ]
            }
         ]
      }
   ]
}
```

Figure 1: Dataset Overview.

3.2 Data Preprocessing

3.2.1 LSTM

Initially, textual components like context, questions, and answers are tokenized, breaking them into discrete units such as words or subwords. This tokenization ensures uniformity across sequences. Subsequently, padding ensures all sequences are of equal length by appending a special token to shorter sequences. Word embeddings are then utilized to convert tokenized words into dense vector representations, effectively capturing semantic relationships. These embeddings aid the model in understanding the contextual meaning of the text.

Following this, input-output pairs are established for LSTM training. Each input sequence comprises a passage-question pair, while the output sequence contains the corresponding answer span or a marker indicating the absence of an answer. Batching optimizes training efficiency by grouping these pairs for simultaneous processing. Tokenized words and answer spans are then converted into numerical representations suitable for LSTM input. Finally, the dataset is partitioned into training, validation, and test sets to accurately assess model performance and prevent overfitting.

3.2.2 BERT

Firstly, tokenization is applied to segment the input context and question into tokens using the same tokenizer employed during the pre-training of the BERT model. Special tokens such as [CLS] to mark the beginning and [SEP] to denote separation are added, along with [CLS] tokens inserted at the beginning of the answer span. Secondly, padding is implemented to standardize the length of tokenized sequences, with shorter sequences padded using a special token ([PAD]) and longer sequences possibly truncated to meet the model's maximum input sequence length. Attention masks are then generated to differentiate between actual words and padding tokens, facilitating the model's focus on relevant tokens during training and inference. Additionally, segment IDs are assigned to distinguish input sequences from different segments, enabling the model to understand the context-question relationship.

Furthermore, the preprocessing pipeline includes the labeling of answer spans within the tokenized context, identifying the start and end positions of the answer text, and creating label vectors to signify the answer span. The preprocessed data is formatted according to the BERT model's input requirements, comprising tokenized input sequences, attention masks, segment IDs, and answer span labels. Finally, the dataset is divided into training, validation, and testing subsets to facilitate model training and evaluation.

3.3 Model Implementation and Fine Tuning

3.3.1 LSTM Experiment

In our research, we explored the effectiveness of the Bi-LSTM network in the context of question-answering tasks, specifically using the SQuAD 2.0 dataset. We developed an LSTM-based model augmented with dropout layers to mitigate overfitting, crucial for managing the model's complexity and enhancing its generalization capabilities. The model architecture consisted of an embedding layer, a bidirectional LSTM, and a linear layer to predict the start and end positions of answers within the context passages.

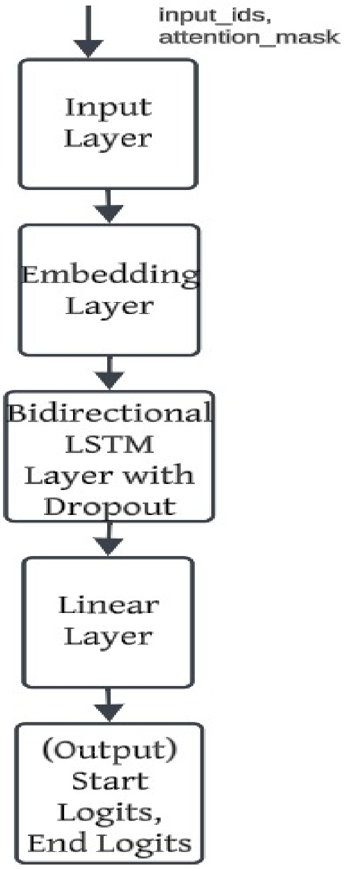

input_ids,
attention_mask
Input Layer
Embedding Layer
Bidirectional LSTM Layer with Dropout
Linear Layer
(Output) Start Logits, End Logits

Figure 2: LSTM Architecture

To facilitate training, we utilized a custom collation function for batching, which dynamically padded sequences to the longest sequence in each batch, ensuring efficient batch processing without fixed sequence length constraints. The training process involved optimizing the model using the Adam optimizer with an initial learning rate set through empirical tuning. Loss was computed separately for the start and end positions of the answers and averaged to update the model weights. Additionally, we employed a learning rate scheduler to reduce the learning rate when the training plateaued, optimizing the training phase for better convergence. Throughout the experiments, the model's performance was quantitatively assessed by tracking the loss and accuracy, providing insights into the effectiveness of LSTMs in handling the complexities of question-answering tasks. This experimentation is useful to study the performance of the model where the attention mechanism is not present. Figure 3 shows the architecture of LSTM-Pipeline.

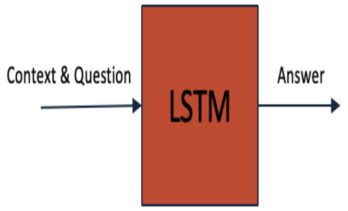


Figure 3: LSTM Pipeline

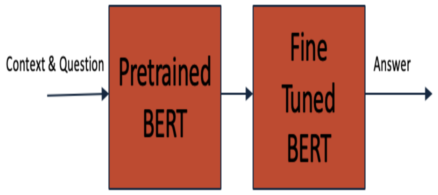


Figure 4: BERT Pipeline

In our experimentation with Bi-LSTM for question answering, we initiated the process with comprehensive data preprocessing, a vital step for preparing the dataset for effective model training. Following this, we implemented a Bi-LSTM model, chosen for its capacity to study the model which doesn't contain an attention mechanism. We then proceeded to the model training phase, where our LSTM model was exposed to various question-answer pairings, teaching it to predict answers based on the input context and question. After training over 20 epochs, we thoroughly evaluated the model.

3.3.2 BERT Experiment

The BERT-base model employs a transformer architecture with 12 layers, 768 hidden units, and 12 attention heads, utilizing pre-training on a large corpus of text to generate contextualized embeddings for diverse NLP tasks. Figure 4 shoes the pipeline for BERT experimentation. We followed a strict protocol in our study to apply a pre-trained BERT-base-cased model which we fetched from hugging face and explored its inherent abilities to answer questions without changing the attention heads considering default 12 attention heads. Our approach consisted of a set of organized phases that started with data normalization. To guarantee consistency in input data, we refined textual content by removing articles, punctuation, and unnecessary whitespace. Our evaluation metrics relied on this normalization in two ways: first, to determine the exact match score—which measures the exact correspondence between the model's predictions and the ground truth—and second, to determine the F1 score, which measures the balance between precision and recall.

We systematically tracked the model's learning progress by recording loss and evaluated the model across 3 epochs. The model underwent rigorous training and validation cycles, where its performance on predicting answer spans was closely monitored and optimized. Crucially, model checkpoints were saved after each epoch, allowing us to preserve the state of the model at each stage of training. This careful monitoring and preservation of training stages provided valuable insights into the model's development and ensured the reproducibility of our results.

3.3.3 Attention Heads Experiment

In our pursuit to study the impact of varying the attention heads on model performance and to check if there could be any other optimal no of attention heads in the architecture of BERT for question answering, we conducted a series of experiments utilizing Optuna, a hyperparameter optimization framework. Figure 5 shows the pipeline for Bert with attention heads experimentation.

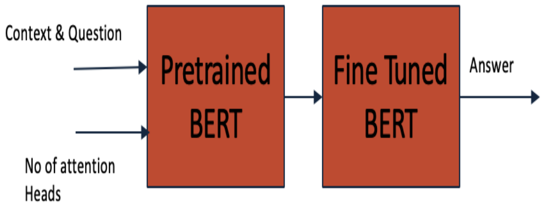


Figure 5: Attention Heads Pipeline

The objective was to discern how varying the attention heads impacts the models' performance and to find the optimal number of attention heads that would yield the highest accuracy. By trialing different configurations, ranging from a narrower focus of 4 to an expansive 16 attention heads, we trained and evaluated each version of the model under identical conditions. The number of attention heads in models like BERT needs to evenly divide the hidden size to ensure that each head's attention computations have consistent dimensions. This alignment allows the model to parallelize learning across the heads efficiently and simplifies the overall architecture, promoting both computational efficiency and easier model maintenance. The experimentation was grounded in the hypothesis that a specific arrangement of attention heads could strike a balance between model complexity and performance. After rigorous testing and validation over multiple epochs, the results were aggregated, guiding us toward an evidence-based decision on the ideal number of attention heads.

3.3.4 Unanswerable Questions Experimentation

BERT should output a prediction that indicates there is no answer available in the given context. In practical terms, this often translates to the model predicting that the start and end positions of the answer span are both at index 0, which typically corresponds to the special [CLS] token in BERT's input sequence. This is because, during training, unanswerable questions are typically labeled with the start and end positions set to 0. When evaluating the model, if it assigns the highest probability to the [CLS] token for both the start and end positions, it's effectively indicating that it does not find a valid answer within the passage.

We first seperated out unanswerable questions from the dataset using "is_impossible" flag, and created dataloader for the same. We used a thorough and discriminating methodology during the evaluation process. If our model predicts start and end logits as "0" then it is considered it has "no answer" and prediction is correct. In order to assess the model's confidence in predicting "no answer," we used a strict threshold. The prediction was deemed to have "no answer" if the model's calculated probability of the [CLS] token serving as both the beginning and the end of an answer was greater than this threshold.

3.4 Results

3.4.1 Evaluation Metrics

- **Exact Match**: The Exact Match (EM) score for question-answering tasks is defined as the proportion of predictions that exactly match any one of the ground truth answers. It is a binary metric that gives a point for a complete match and zero for anything less, regardless of how close the prediction might be. This score is strict because it requires the model's predicted answer to be identical to the ground truth after normalizing for factors like case, whitespaces, and punctuation. It is often expressed as a percentage, representing the total number of perfect matches over the total number of questions posed to the model.

- **F1 score**: Serves as a measure of a model's performance in question answering scenarios where the answer can be a span within a passage. It is computed as the harmonic mean of precision and recall. Precision represents the proportion of correctly identified tokens (words) in the predicted span relative to the total number of tokens in the predicted span, while recall signifies the proportion of correctly identified tokens in the predicted span relative to the total number of tokens in the ground truth span. For each individual prediction, the F1 score is calculated by taking twice the product of precision and recall and dividing it by the sum of precision and recall. The final F1 score for the model is derived as the average of these scores across all questions, providing a comprehensive assessment of the model's ability to accurately identify answer spans within passages.

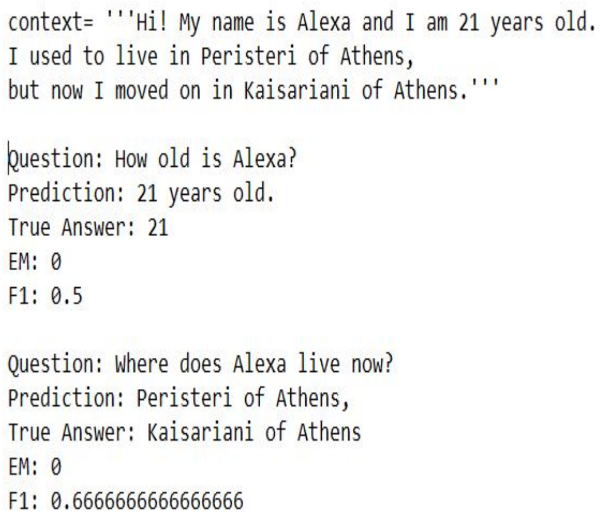
3.4.2 Model Output

The BERT model is provided with both context and question inputs, following which it predicts an answer. Subsequently, we display the true answer, Exact Match, and F1 score.

Figure 6 provides model output for our custom testing.

3.4.3 Analysis of Results

For **LSTM Experimentation** as dicussed in 3.3.1 we attained an Accuracy of 47% using F1 metric and Exact Match score of 34% on validation set over 20 epochs with no attention mechanisms in the model. Overall, the LSTM model shows



```
context= '''Hi! My name is Alexa and I am 21 years old.
I used to live in Peristeri of Athens,
but now I moved on in Kaisariani of Athens.'''

Question: How old is Alexa?
Prediction: 21 years old.
True Answer: 21
EM: 0
F1: 0.5

Question: Where does Alexa live now?
Prediction: Peristeri of Athens,
True Answer: Kaisariani of Athens
EM: 0
F1: 0.6666666666666666
```

Figure 6: BERT Model Output

a positive learning trajectory, yet the accuracy achieved by the 20th epoch suggests that there is room for improvement. Figure 7 shows LSTMs' model performance over the training period of 20 epochs.

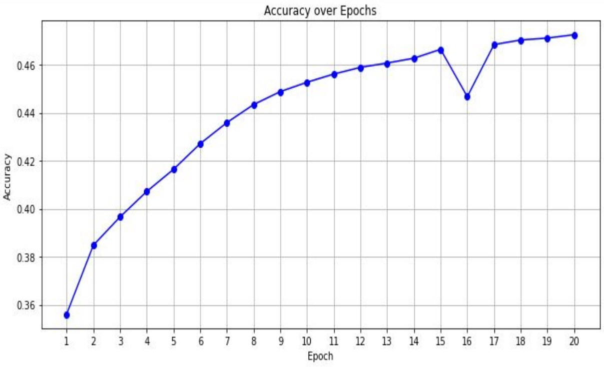


Figure 7: LSTM Accuracy over 20 epochs

For **BERT Experimentation** as discussed in 3.3.2 we attained an accuracy of 73% using F1 metric and Exact Match score of 59% on validation set over 3 epochs with 12 attention heads in the model.

The results state that the model even though trained over 3 epochs when compared 20 epochs of the LSTM model, better outperformed the LSTM model because of multi-head attention mechanisms in bert as it helps better capture nuances and language from the context of the question. Therefore, our hypothesis by these experiments BERT with its bi-directional contextual understanding and attention mechanisms will outperform LSTM.

Figure 8 & 9 shows pre-trained BERT-base-cased models' performance over the period of 3 epochs.

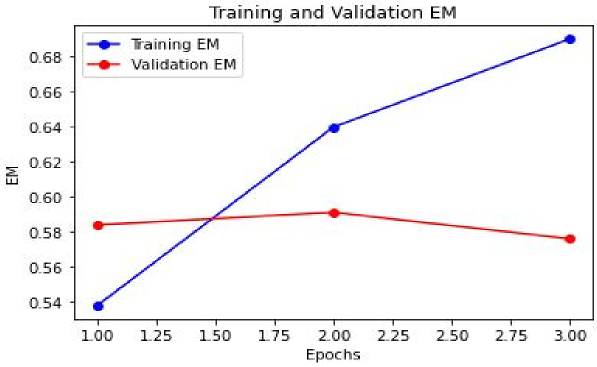


Figure 8: BERT EM score with 12 attention heads

For **Attention Heads Experimentation** as discussed in 3.3.3 As we introduced the BERT model with attention heads, we witnessed a substantial performance leap. With just 4 attention heads, BERT's scores improve significantly, achieving an F1 of 0.65 and an Exact Match of 0.50. This jump highlights the pivotal role of attention heads in capturing the complexities of language and context within the dataset. Further incrementing the number of attention heads to 8 and then 12 leads to a continued uptrend in performance, peaking at 12 heads with an F1 score of 0.73 and an Exact Match of 0.59. This trend underlines the enhanced capacity of the model to focus on relevant segments of the text, thereby improving its understanding and the accuracy of its predictions.

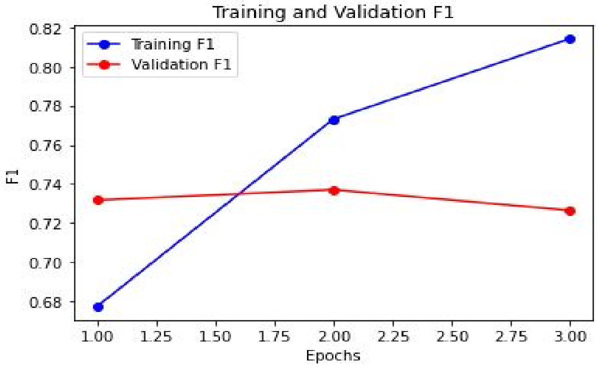


Figure 9: BERT F1 score with 12 attention head

However, when the number of attention heads is increased to 16, there is a slight regression in performance, with a drop in both F1 and Exact Match scores. This decrease may be indicative of the model reaching a threshold beyond which additional attention heads no longer contribute to performance gains but instead introduce unnecessary complexity. This complexity could potentially lead to overfitting, where the model becomes too specialized to the training data, impairing its ability to generalize to new data. The optimal number of attention heads, in this case, appears to be 12, offering a balance between model depth and performance without incurring the diminishing returns of over-complexity.

For **Unanswerable Questions Experimentation**: We Utilized Optuna Hyperparameter optimization library to calculate the threshold(which is the confidence score in predicting no-answer) as mentioned in the section 3.3.4. We ran the model over a validation set for 3 epochs utilizing optuna and achieved an accuracy score of 69% with the most optimal threshold being 0.46.

Table 1 presents cumulative results, indicating the model, number of attention heads, and evaluation metrics F1 and Exact Match Scores, with the BERT model with 12 attention heads performing the best.

MODEL	**No of Attention Heads**	**F1 SCORE**	**Exact Match**
LSTM	0	0.47	0.34
Pretrained BERT (base)	4	0.65	0.50
Pretrained BERT (base)	8	0.69	0.54
Pretrained BERT (base)	12	0.73	0.59
Pretrained BERT (base)	16	0.70	0.55

Table 1: Comparison of model performance with different number of attention heads.

4 Conclusion and Future Scope

Based on our project findings, it is evident that LSTM, devoid of any attention heads, exhibited the least performance among all models evaluated. Subsequently, BERT, equipped with attention heads, demonstrated superior performance, showcasing an enhanced understanding of language nuances. Notably, as the number of attention heads increased, the model's performance exhibited improvement, indicative of its increased ability to capture intricate linguistic patterns. However, beyond a certain threshold, a diminishing trend in model accuracy was observed, signaling the onset of overfitting. Concurrently, the increase in the number of attention heads was accompanied by escalating computational complexity, highlighting a trade-off between model performance and computational efficiency. Thus, our conclusion underscores the critical need for optimizing attention mechanisms to strike a balance between model effectiveness and computational resources in complex natural language processing tasks.

In the future, there is scope to investigate attention mechanisms to mitigate the onset of overfitting observed with increasing attention heads. This could involve exploring alternative attention mechanisms or architectures that maintain or enhance performance while reducing computational complexity.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Daniel Jurafsky and James Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.