Milestone 2 Report

CSEN1076: Natural Language Processing and Information Retrieval

Team Members:

Mazen Soliman (52-2735)
Mohamed Shamekh (52-0989)

Supervised By: Mayar Osama

1 Introduction

The purpose of this milestone was to implement a shallow neural network for information retrieval and question-answering tasks. We used the SQuAD dataset which contains a context and a question with an answer and answer's start index that can be used to retrieve the answer from the context.

2 Approach 1: Transformer-Based QA Model

2.1 Methodology

This approach focuses on utilizing the Attention mechanism in the Transformer model specifically Transformer encoder and cross attention mechanism to build a QA model that predicts the exact start and end index of the question's answer.

2.1.1 Word Embedding

First, we use an embedding layer that is learned during training with the number of embeddings equal to the length of the vocab in the training dataset with addition of two embeddings one for unknown tokens and another for padding.

2.1.2 Encoder Mechanism

We used the standard **Transformer Encoder** architecture as described in [6], illustrated in Figure 1. First, we pass the embeddings of both the context and the question, each added to their positional embedding to provide the model with information about the positions of the tokens. Where the positional embedding can be calculated as follows:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{1000^{\frac{2i}{d}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{1000^{\frac{2i}{d}}}\right)$$

Where:

- pos is the position of the token in the sequence.
- i is the specific dimension within the embedding vector.
- d is the dimensionality of the embeddings.
- $10000^{\frac{2i}{d}}$ ensures that each dimension of the positional embedding has a different frequency.

Then, **self-attention** is applied separately to the context and question embeddings. This mechanism captures the relationships between each word and its surrounding words, enabling the model to encode richer contextual information. The output of the self-attention layer is then added to its input and normalized. This normalized result is passed through a feed-forward neural network to extract additional features. The output of this feed-forward layer is again added to its input and normalized. The final outputs are the encoded representations of the context and the question.

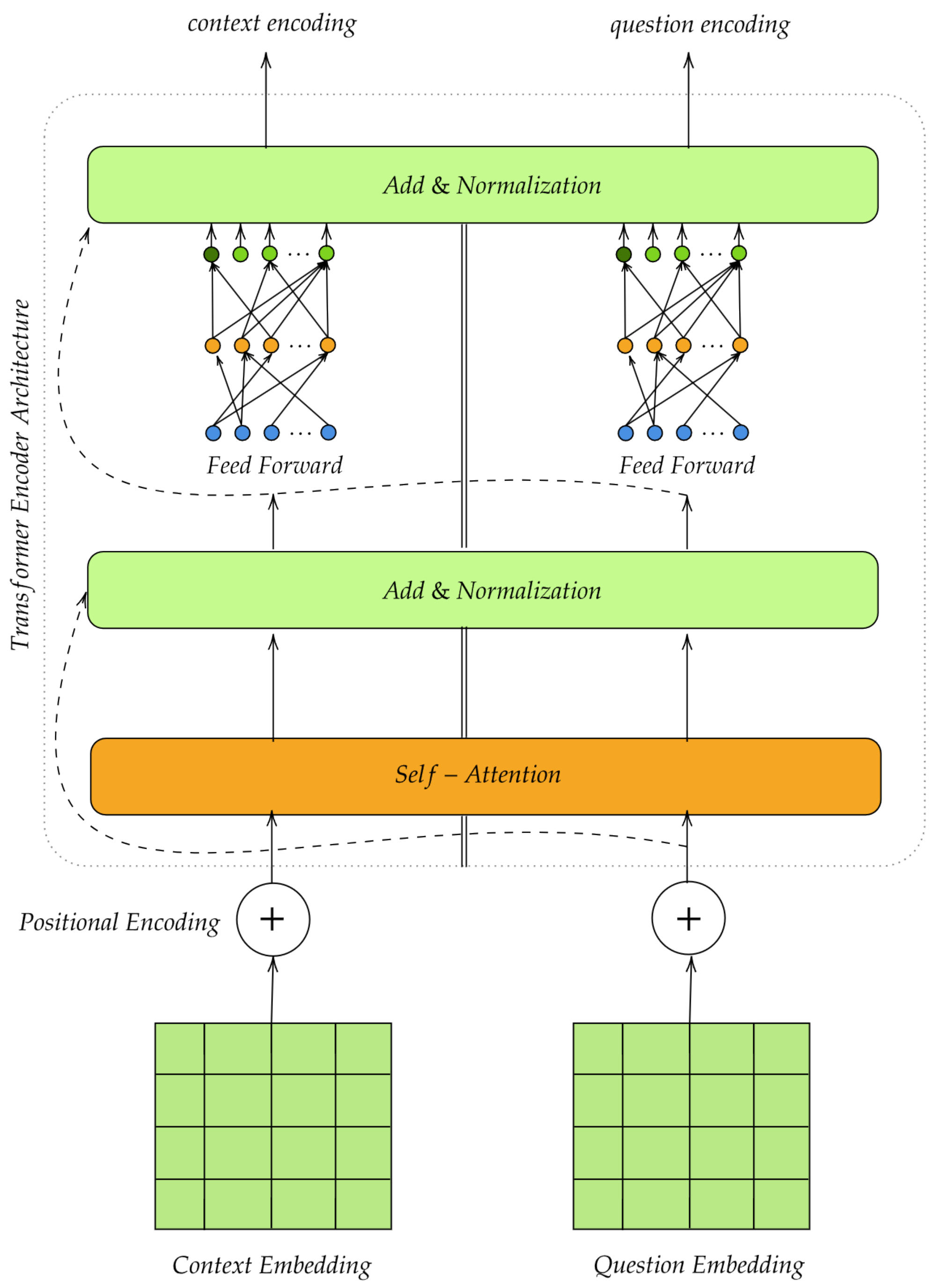

Figure 1: Transformer Encoder Architecture

2.1.3 Model Architecture

As shown Figure 2, Our model begins by embedding both the context and question token sequences and adding positional embedding. Then, it builds padding masks to ignore padded positions during attention. The model splits its transformer layers into two halves: the "pre-cross" layers which represented by the first encoder layer that apply standard self-attention and feed-forward blocks separately to the context and question, enriching each with intra-sequence context. Next, a cross-attention layer lets each context token attend over the question representations (followed by dropout and layer normalization), injecting question-aware information into the context. After cross-attention, the "post-cross" layers represented as the second encoder layer again perform self-attention and feed-forward processing on the context alone, refining these integrated representations. Finally, a small three-layer feed-forward head, layer normalization, and dropout—projects each contextualized token into a two-dimensional logit space, from which the model slices out start and end logits for span prediction.

2.1.4 Model Training

We trained our model with 0.1 dropout value and also with 512 hidden state size and embedding dimension of 300. We ran our model for 30 epochs with batch size 16 and took a checkpoint after each 5 epochs. The model was optimised using the ADAM optimiser with a learning rate of 0.001. We calculated the loss using the cross-entropy loss function with weighted ratio for the start and end index loss.

Note

For post-processing, we convert the start and end indices into a string by extracting the corresponding substring from the context, which constitutes the model's predicted answer.

2.1.5 Model Evaluation

In Table 1, we display the performance of the model after each 5 epochs. Where we trained our model over 17722 samples from our training set with maximum length 400 and with 12533 unique contexts and evaluated our model over 735 samples from the validation set (different from the training set) using two F1 scores: one based on correct predicted text in the predicted answer compared to the ground truth and another based on the start and end index from scikit-learn.

Epoch Number	F1 Score	EVM Score	Start F1 Score	End F1 Score
5 **Epochs**	15.966%	5.578%	4.619%	12.524%
10 **Epochs**	13.770%	3.265%	3.903%	8.422%
15 **Epochs**	13.330%	3.809%	4.128%	8.591%
20 **Epochs**	14.081%	4.081%	4.656%	9.017%
25 **Epochs**	13.997%	4.081%	3.997%	7.944%
30 **Epochs**	13.426%	3.809%	3.552%	8.248%

Table 1: F1 & EVM Scores

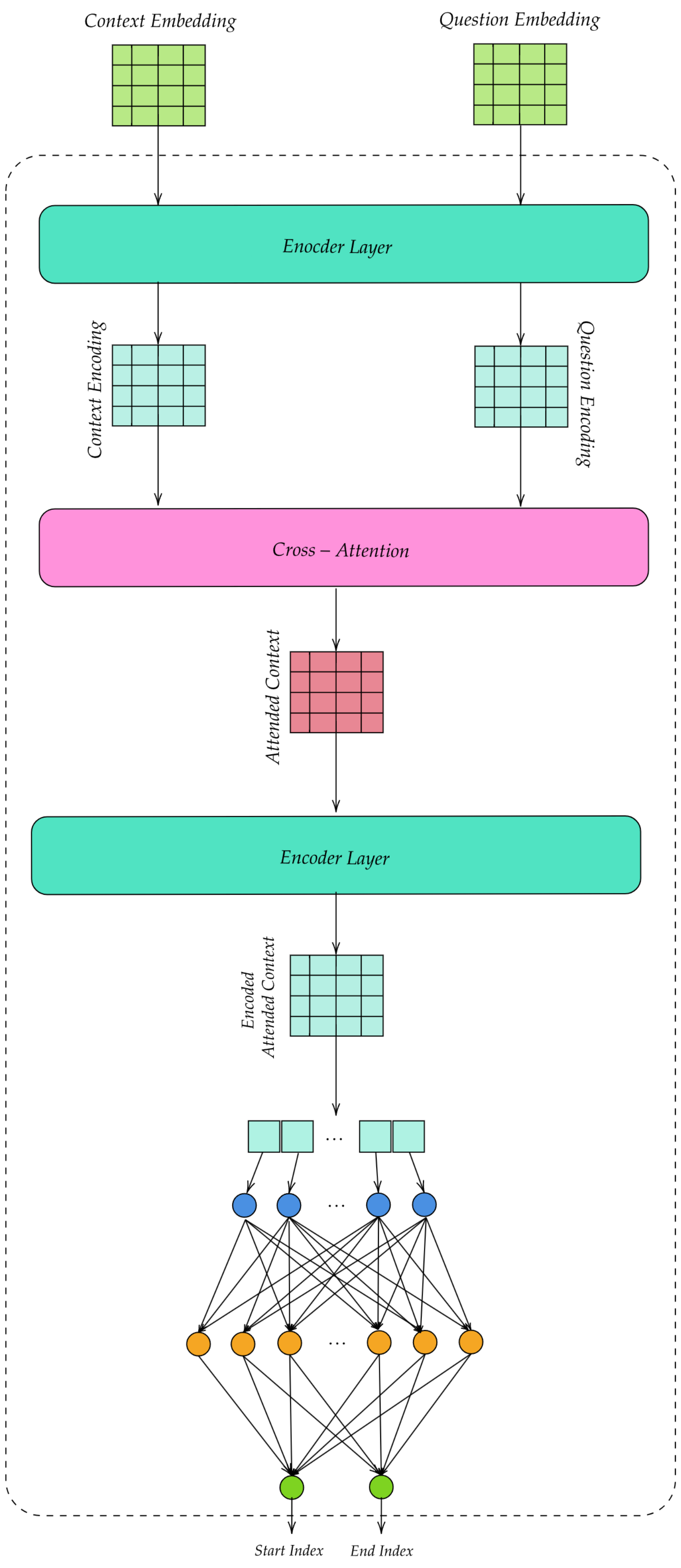

Figure 2: QA Transformer-Based Model Architecture

Observation

The model's F1 score is quite low, and we can observe that both the start and end F1 scores are significantly lower than the text-based F1 score. This could be due to the model generating excessively long answers, resulting in a large gap between the start and end indices. Hence, the predicted start and end indices are further away from the ground-truth but the correct answer could be within the retrieved text.

3 Approach 2: Simplified Transformer-Based QA Model

3.1 Methodology

We decided to simplify our model from unnecessary components that does not contribute much to solving our task where the process is made through trial and error.

3.1.1 Word Embedding

We represented the textual input from the context and question into a representative format that can be understood by our model. In which we utilized GloVe [4], which is an unsupervised learning algorithm to obtain vector representations for words.

Also, we noticed that not all words that found in the context are represented as vectors using GloVe. Hence, we represent unknown words with random vector representation and we added an embedding layer with random weights initially that is finetuned during training to capture the embeddings of these unknown word's representation.

Note

There was no much pre-processing required as we wanted to keep the nature of our context to be able to predict start and end indices of tokens in the context. Also, we sometimes use the lemma of words to get their embeddings if the exact word did not exist in our vocab but it's lemma.

3.1.2 Model Architecture

We simplified our architecture in section 2, as shown in Figure 3. We removed the pre-encoding layer, as it did not significantly contribute to the learning process and could negatively alter the word embedding representations—potentially impacting the effectiveness of cross-attention, which relies on attending from the context to the question. Additionally, we added a ReLU activation function in the three-layer perceptron to add non-linearity.

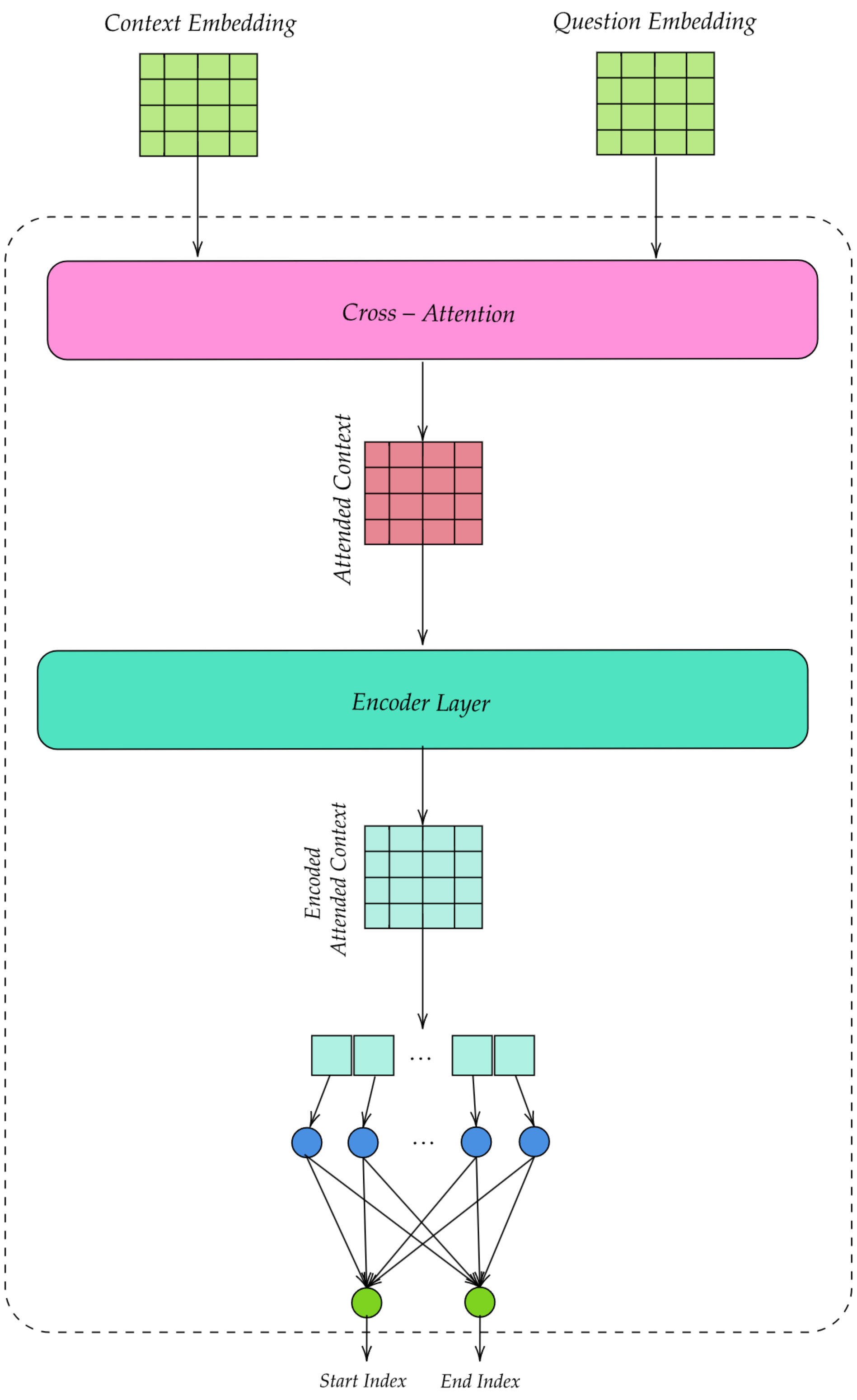

Figure 3: Simplified QA Transformer Based Model Architecture

3.1.3 Model Training

We used similar hyperparameters as in section 2 with minor adjustments to the loss estimation which we also included not only the start and end loss but also the length of the answer as we noticed that the model frequently provide very long answers rather than being specific to the required answer. Also, the encoder phase consists of two **Transformer Encoder** layers effectively encode the output of the cross-attention layer.

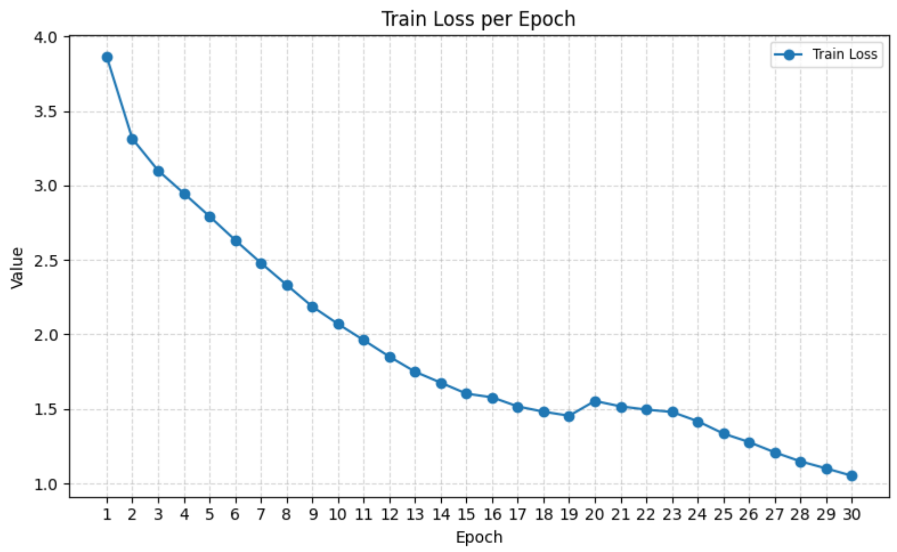

Figure 4: Training loss is declining throughout training which indicate the model is learning.

3.1.4 Model Evaluation

We used similar evaluation metrics as in section 2, where the result are shown in Table 2.

Epoch Number	F1 Score	EVM Score	Start F1 Score	End F1 Score
5 **Epochs**	21.570%	7.074%	8.301%	12.451%
10 **Epochs**	21.883%	7.482%	8.712%	13.879%
15 **Epochs**	19.558%	6.530%	8.044%	11.051%
20 **Epochs**	17.057%	4.897%	7.329%	9.012%
25 **Epochs**	17.232%	5.306%	6.624%	11.272%
30 **Epochs**	17.375%	5.034%	6.609%	11.210%

Table 2: F1 & EVM Scores

Observation

We notice that as the number of epochs increase the model performance decreases which could be due to overfitting.

3.1.5 Limitations

Even though our model performed better than the previous approach still it produces very long answers. which could be due to using considerably small dataset which does not help much in making the model generalize.

3.1.6 Visualization

Here, we provide different analysis for the output of our model. In Figure 5, we can see that our model favors long length answers with respect to the ground truth as their fairly lots of predicted answer's length between 20 and 80 while the longest answer in the validation dataset is 16.

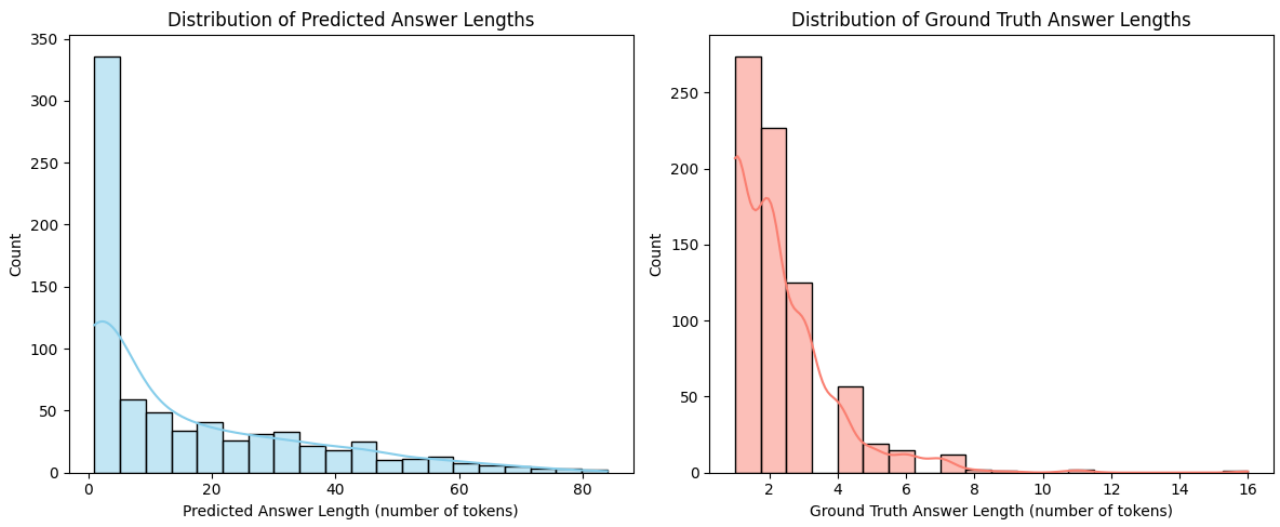

Figure 5: Score distribution

In Figure 6, we observe that their is similar F1 score for different answer lengths with slightly higher value for long answer lengths, this could be due to the model generating very long answers.

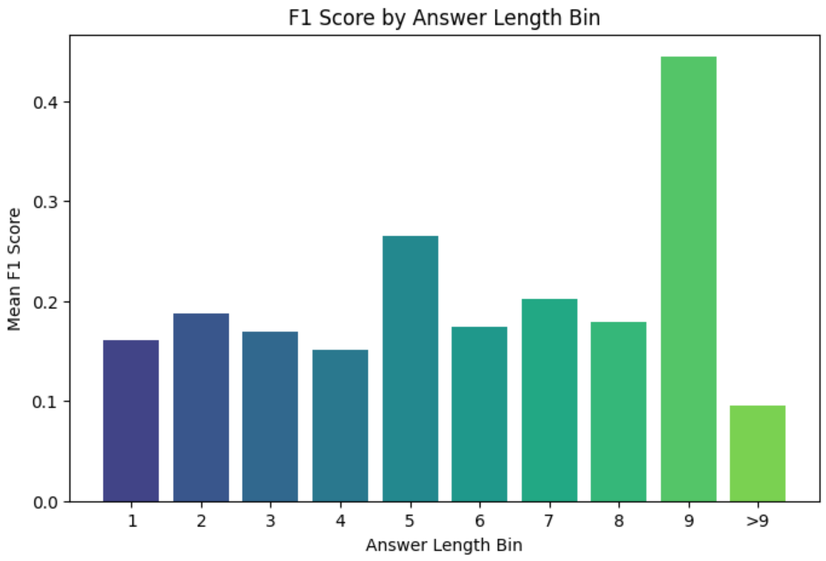

Figure 6: F1 score with respect to answer length bins (length 1, 2, 3, ..., >9)

Finally, the figure Figure 6 represent the F1 score of our model with respect to different types of questions where we can notice that the model have higher performance with "when" types of questions with respect to other types of questions.

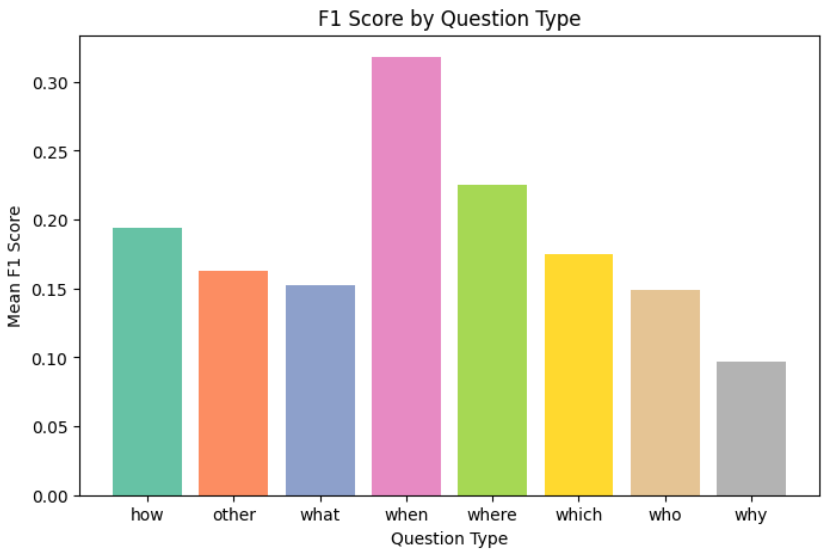

Figure 7: Performance Variation (F1 score) with Question Type (Categories: what, who, when, how, which, where, why, other)

4 Approach 3: Co-Attention BiLSTM QA Model

4.1 Methodology

We have investigated multiple neural network architectures either as a language model or information retrieval where the one that we implemented is based on [3].

4.1.1 Word Embedding

Similar to section 3, we use GloVe to represent word embeddings with additional embedding layer to learn the embedding of words that are not present in GloVe.

4.1.2 Encoder Mechanism

We used the Co-Attention mechanism discussed in [3]. As shown in Figure 9, we first compute the affinity matrix which contains the affinity scores of all pairs of passage and question words: $A = P^T Q$. The affinity matrix then convoluted using $1D$ Gaussian Convolution so that words with high attention are smoothed for words near it with kernel size 11. The concept comes from the fact that words with high attention does not always correspond to the answer. Hence, by using Gaussian Convolution, the kernel would determine the attention of a central word by adding the weighted attention of all its neighbors words together. Therefore, words that are near words with high attention would have high attention too as shown in Figure 8.

4.1.3 Decoder Mechanism

Now, the co-attention encoder output is decoded using double-layered Bi-direction LSTM (BiLSTM). Which allows to decode the joint representation of the question and the context, so that we can compute the probability distributions of *span start* and *span end*. The steps in the decoder architecture are shown in Figure 10.

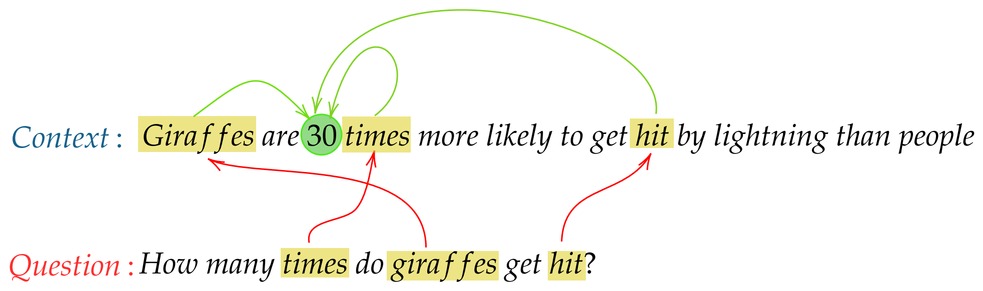
Context : Giraffes are 30 times more likely to get hit by lightning than people
Question : How many times do giraffes get hit?

Figure 8: Attention Sharing

4.1.4 Model Training

We trained our model with 0.3 dropout value and also with 128 hidden state size and embedding dimension of 300. We ran our model for 15 epochs with batch size 32 and took a checkpoint after each 5 epochs. The model was optimised using the ADAM optimiser with a learning rate of 0.001. We calculated the loss using cross entropy loss function with weighted ratio for the start and end index loss where higher ratio is assigned to the start index as the end index is just dependent of the start index. Also, we included the difference in length of answer in our loss from the ground truth as we observed the model tend to output same start and end index.

Moreover, we attempted different approaches to enhance our model performance such as including the difference in cosine similarity between the predicted answer and the ground truth answer and also using gradient clipping to prevent the exploding gradient problem but this approaches did not effect the performance of our model very much.

4.1.5 Model Evaluation

The output of after each 5 epochs is shown in Table 3. Where we trained our model over 17722 samples from our training set with maximum length 400 and evaluated our model over 735 samples from the validation set (different from the training set).

Epoch Number	F1 Score	EVM Score
5 **Epochs**	15.616%	6.802%
10 **Epochs**	15.142%	7.347%
15 **Epochs**	16.124%	7.755%

Table 3: F1 & EVM Scores

4.1.6 Limitations

We noticed that in [3], they were able to achieve ($\approx$ 60) F1 Score which could be the reason that they trained over large amount of training data (87k samples). Also, they did not mention the number of layers in their BiLSTM in which we only use 1 layer per each BiLSTM model where a deeper architecture could indeed capture more features that allow the model to generalize and produce better results. Also, training our model on larger dataset yields better performance with a dynamic convolution layer over 97882 samples lead to 21.39 F1 score and 10.48 EVM over 5 epochs with training loss reaching 2.3961 where the model showed the ability to still learn if we kept on training the model for more epochs but we stopped it due limited resources and time.

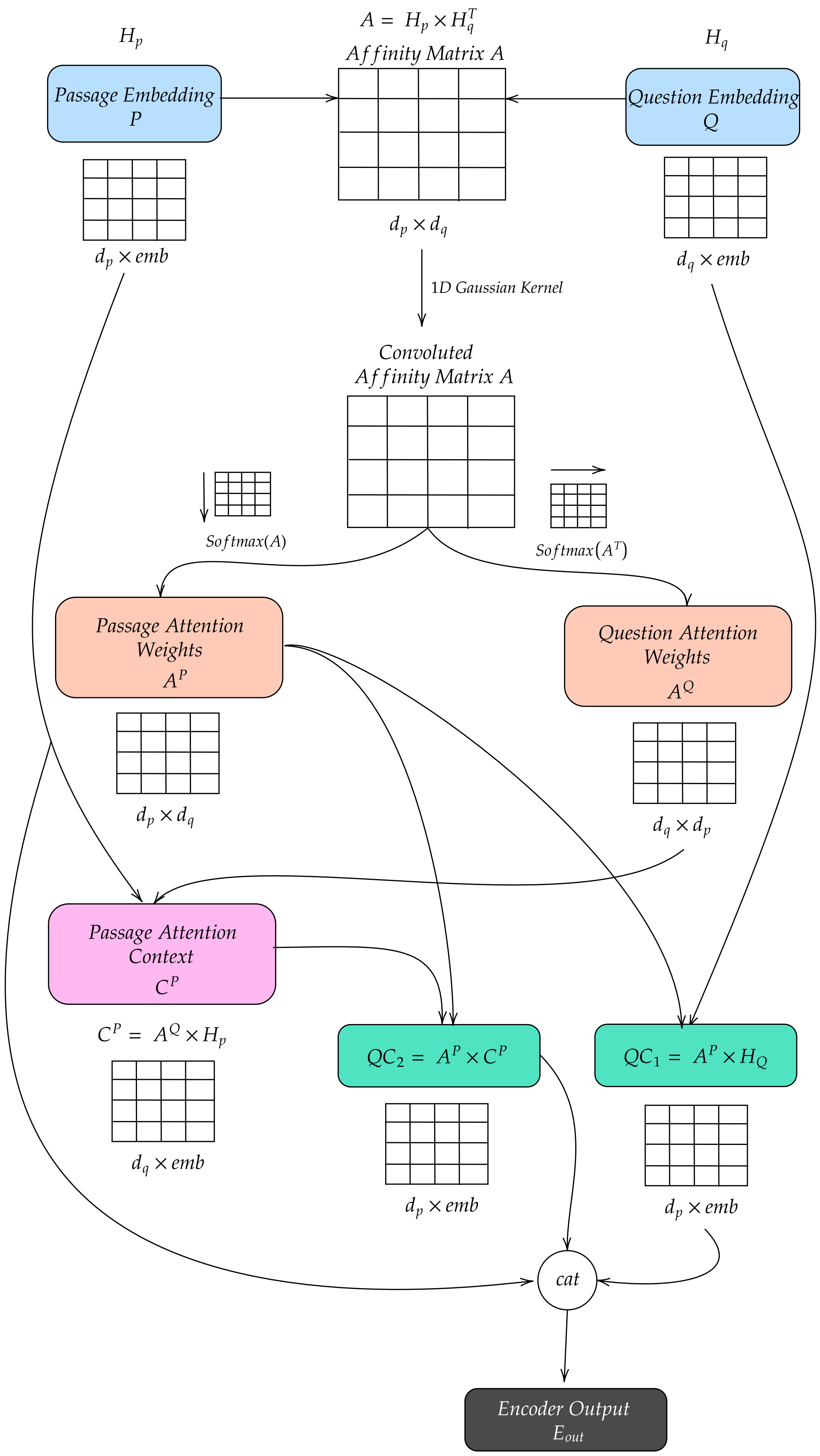

Figure 9: Co-Attention Encoder Architecture

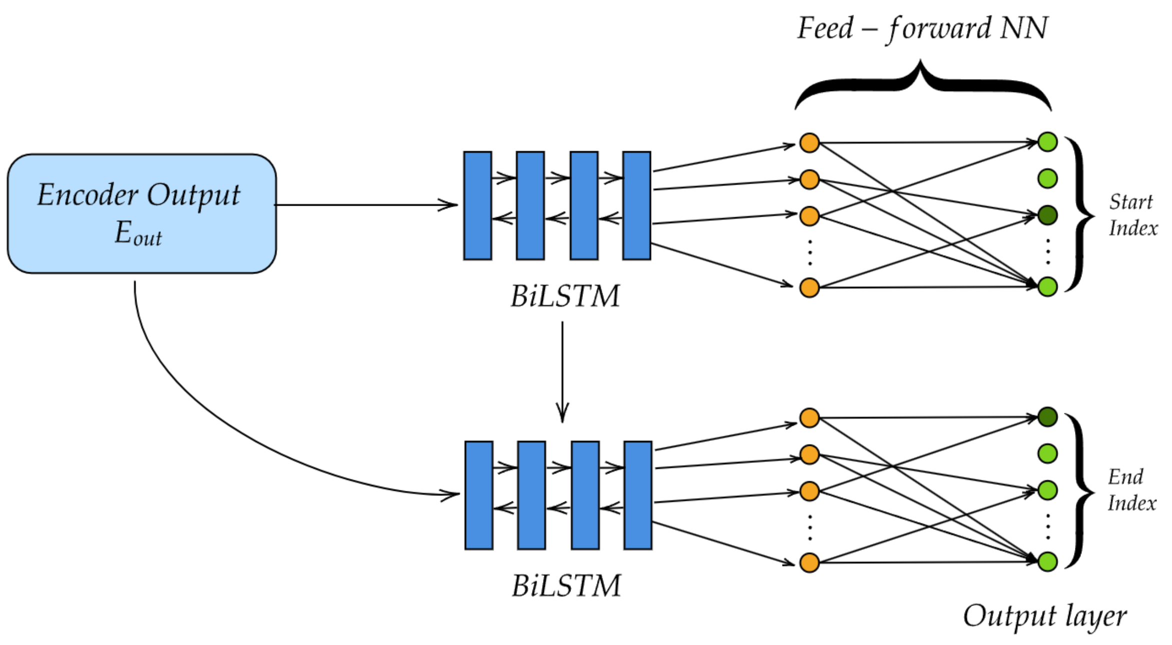

Figure 10: Decoder Architecture

4.1.7 Visualization

Here, we provide different visuals for the output of our model. In Figure 11, we can see that our model favors low length answer than long answer texts. This could be due to that large amount of answers are between 1 to 3 in size.

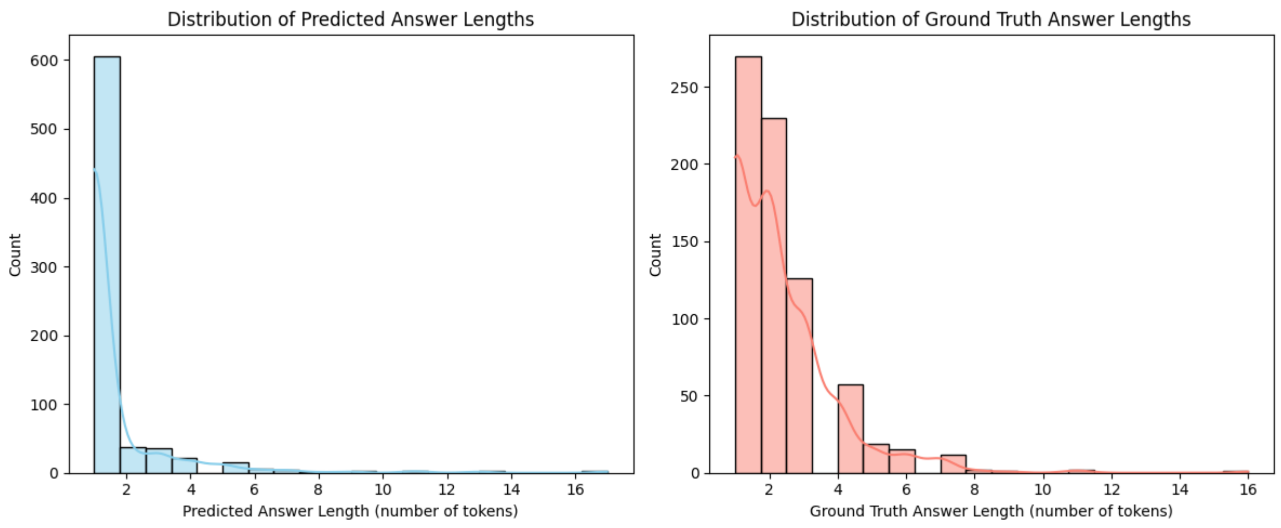

Figure 11: Score distribution

In Figure 12, we observe that their is similar F1 score for different answer lengths with slightly higher value for low answer lengths, this is due to the dataset containing large number of answers with small lengths and our model being shallow.

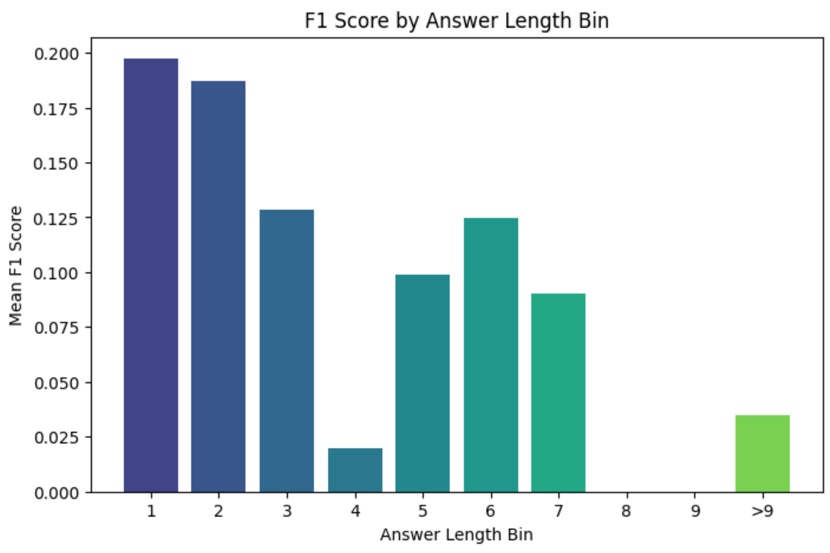

Figure 12: F1 score with respect to answer length bins (length 1, 2, 3, ..., >9)

Finally, the figure Figure 13 represent the F1 score of our model with respect to different types of questions where we can notice that the model have higher performance with "when" types of questions with respect to other types of questions.

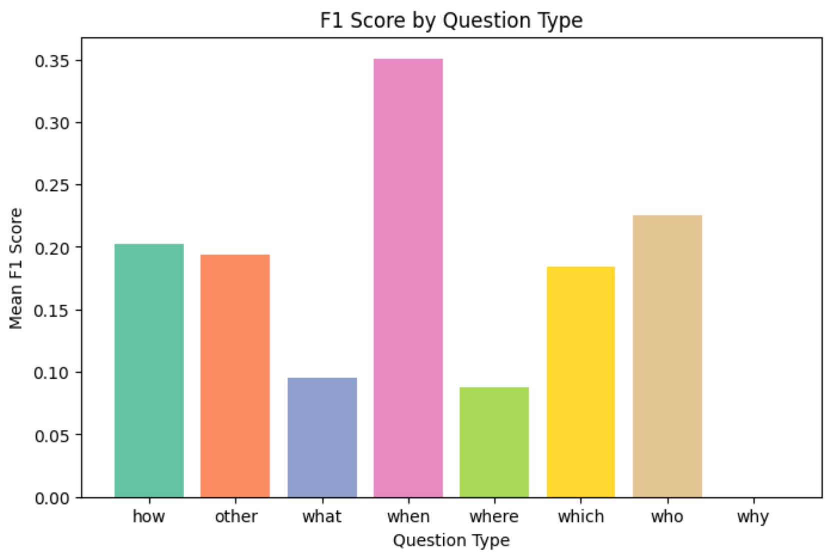

Figure 13: Performance Variation (F1 score) with Question Type (Categories: what, who, when, how, which, where, why, other)

References

[1] Rsocher Cxiong, Vzhong. Glove: Global vectors for word representation, 2024. https://ar5iv.labs.arxiv.org/html/1611.01604.

[2] DataCamp. Transformer models with pytorch. https://www.datacamp.com/courses/transformer-models-with-pytorch.

[3] Sanyam Goyalk, Sriraman. Neural network-based question answering system, 2018. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2760290.pdf.

[4] Christopher D.Manning Jeffrey Pennington, Richard Socher. Glove: Global vectors for word representation, 2018. https://nlp.stanford.edu/projects/glove/.

[5] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad dataset, 2016.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.