Question Answering on SQuAD

Beliz Gunel
Department of Electrical Engineering
Stanford University
bgunel@stanford.edu

Cagan Alkan
Department of Electrical Engineering
Stanford University
calkan@stanford.edu

Abstract

In this paper, we implement Bidirectional Attention Flow(BiDAF) model [1] and make several adjustments to modeling layers and hyperparameters to increase the performance. Our single model achieves 78.3% F1, 69.2% EM on the test set and has a competitive rank on the class leaderboard. We also have higher EM and F1 scores than the original BiDAF implementation with single model.

1 Introduction

Reading Comprehension is a popular Question Answering task, where the system tries to provide a correct answer to a query about a given context through selecting the span of text in the corresponding paragraph. If successful, this will have many practical applications such as virtual assistants and automated customer service. It can also be used as a measure of how well NLP systems can understand various texts for research studies. In order to facilitate the progress in this task, Stanford NLP group released the Stanford Question Answering Dataset (SQuAD) [2], a reading comprehension dataset which has more than 100K question-answer pairs and their corresponding context paragraphs from Wikipedia. Most successful systems in the SQuAD public leaderboard utilize some sort of attention mechanism to focus on a small portion of the context and summarize it.

In this project, on top of the given baseline, we implement BiDAF network that models query-aware representations of the context paragraph without early summarization. BiDAF utilizes character-level, word-level, and contextual embeddings. We make several modifications to the original BiDAF network to increase the performance which will be described in detail in Section 3. We also report our results along with a visualization of our attention mechanism and distributions of the start and end positions of the predicted answer spans in Section 4.

2 Related Work

Reading comprehension using neural networks has been heavily studied in the past few years as one can see from the SQuAD dataset leaderboard [3], with Reinforced Mnemonic Reader + A2D ensemble model even surpassing human level performance based on the EM metric. Most high-performing models use neural attention mechanism to combine the representations for the context and the question, which was first introduced in Neural Machine Translation systems. [4] Dynamic Coattention Network [5] is another successful model, which not only uses two-way attention between the context and the question like BiDAF but also attends over representations that are themselves attention outputs. There are also models with simpler attention techniques like self-attention implemented in R-Net [6] and basic dot-product attention.

Using additional features in addition to word vectors such as Part-of-Speech tag, the named entity type and aligned question embedding is another way to improve performance [7]. Implementing different strategies to predict the start and end location of the answer are also commonly investigated. Match-LSTM with Answer Pointer model [8] conditions end prediction on start prediction instead of

predicting each independently. Dynamic Chunk Reader model [9] determines the joint probability distribution of the start and end locations defined as random variables given the context and the question.

3 Approach

3.1 Baseline Model

Our baseline model consists of 4 main layers. In the **word embedding layer**, fixed size GloVe vector representations of words are obtained. The word embeddings are then fed into **RNN encoder layer** which uses a one layer bidirectional GRU to obtain context and question hidden states. Encoder weights are shared between the context and the question. **Attention layer** employs a basic dot-product attention mechanism to combine context and question representations. Blended representations are obtained through concatenating the attention outputs and context hidden states. Finally, **output layer** applies a fully connected layer and two separate softmax layers to the blended representations in order to obtain start and end location distributions over the context. During training, the sum of the cross-entropy losses of start and end locations is used as the loss function. In our baseline, the predictions are made by taking the argmax over start and end distributions independently.

3.2 Bi-Directional Attention Flow with Character Level Embedding

Our proposed model is a slightly modified version of BiDAF network. BiDAF network introduces a bi-directional attention mechanism to obtain query-aware context representations without early summarization. As shown in Figure 1, our architecture consist of 6 main layers that are described below in detail.

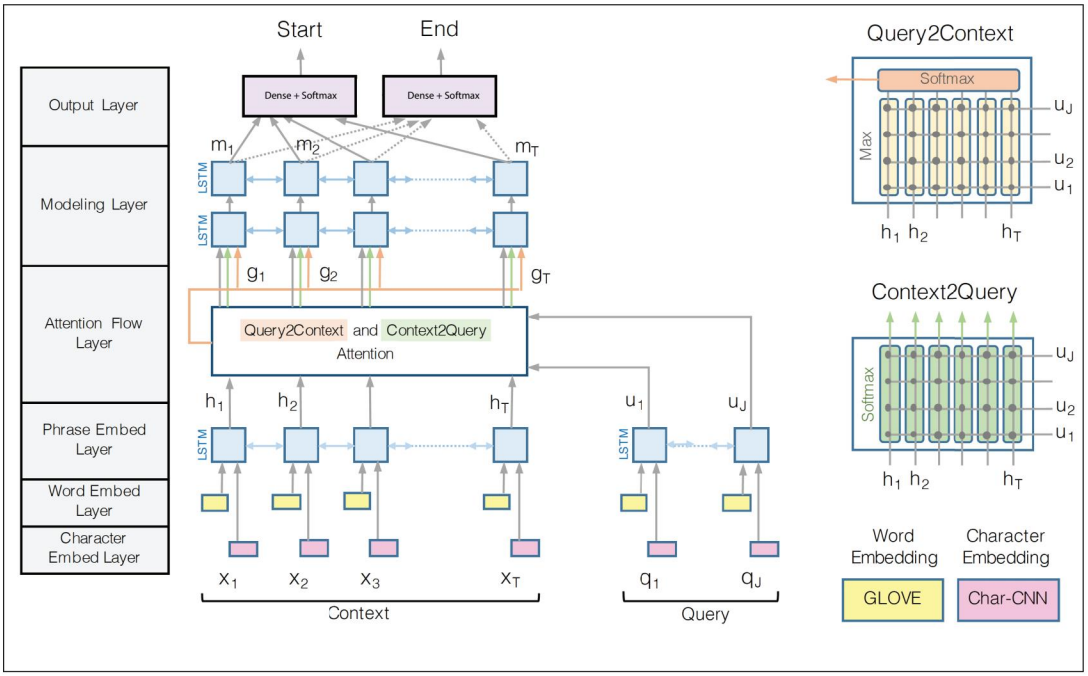

Figure 1: Model architecture.

3.2.1 Word and Character Embedding Layer

We use embedding layers at both word and character level to obtain vector space representations of each word in the context and query. For word embeddings, the context and question are represented

by sequences of d-dimensional pre-trained GloVe embeddings $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbf{R}^d$ and $\mathbf{y}_1, \ldots, \mathbf{y}_M \in \mathbf{R}^d$, respectively. These embeddings are fixed and are not updated during training in our model.

Character embedding layer [10] allows us to capture the morphology of the words better and helps to represent out-of-vocabulary words. Each character in a word has a d_c-dimensional embedding $\mathbf{e}_1, \ldots, \mathbf{e}_L \in \mathbf{R}^{d_c}$ that is randomly initialized in the beginning of training. For each word, these character embeddings are concatenated and passed through a 1-dimensional convolutional layer. The convolutions are applied over word length (L), and embedding size (d_c) acts as the input channel size. Number of convolutional filters, f, which is a hyperparameter, determines the output channel size of the convolutional layer. Output of the convolutional network is then max-pooled over the feature dimension to obtain fixed-size f-dimensional vector representation for each word. We call these representations 'character-level encoding' for each word. Character-level encodings are obtained for both context and question.

Finally, we concatenate word embeddings and character-level encodings to have a hybrid representation of words. Hybrid representations are passed directly to contextual embedding layer. This is different than the implementation in BiDAF paper since they used Highway Networks as an intermediate layer. We wanted contextual embedding layer to work directly with the hybrid representations. In summary, we obtain the hybrid representations $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_N \in \mathbf{R}^{d+f}$ and $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_M \in \mathbf{R}^{d+f}$ for context and question, respectively.

3.2.2 Contextual Embedding Layer

Contextual embedding layer encodes hybrid embeddings of question and context to hidden states. We used 1-layer bi-directional LSTM to capture the temporal interactions between words. The weights of biLSTM are shared between context and question to enrich their representations. Forward and backward hidden states produced by the biLSTM are concatenated to obtain context and question hidden states $\boldsymbol{h}$ and $\boldsymbol{u}$, respectively.

$$\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\} = \text{biLSTM}\big(\{\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_N\}\big)$$
$$\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M\} = \text{biLSTM}\big(\{\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_M\}\big)$$

where $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}] \in \mathbf{R}^{2h}$ for $i = 1, \ldots, N$ and $\mathbf{u}_i = [\overrightarrow{\mathbf{u}_i}; \overleftarrow{\mathbf{u}_i}] \in \mathbf{R}^{2h}$ for $i = 1, \ldots, M$.

3.2.3 Attention Flow Layer

Attention flow layer combines question and context hidden states to obtain question aware context representations. Unlike basic attention, the main idea behind bi-directional attention is that the attention flows in two directions: from question to context and from context to question. Both of these are calculated from a similarity matrix $S \in \mathbf{R}^{N \times M}$ whose entries are computed as:

$$S_{ij} = \boldsymbol{w}^T[\mathbf{c_i}; \mathbf{q_j}; \mathbf{c_i} \circ \mathbf{q_j}]$$

where $\circ$ represents elementwise product and $\boldsymbol{w} \in \mathbf{R}^{6h}$ is a parameter to be learned. S_{ij} indicates the similarity between i-th context word and j-th question word.

For Context to Question (C2Q) attention, we take the softmax over the columns of S and obtain attention distributions a^i over question hidden states for each context word. C2Q attention distribution is then used to take weighted sum of question hidden states to obtain C2Q attention output $\boldsymbol{a}_i$.

$$a^i = \text{softmax}(S_{i,:}) \in \mathbf{R}^M \quad \forall i \in \{1, \ldots, N\}$$
$$\boldsymbol{a}_i = \sum_{j=1}^{M} a^i_j \boldsymbol{u}_j \in \mathbf{R}^{2h} \quad \forall i \in \{1, \ldots, N\}$$

We perform question to context (Q2C) attention by first calculating the maximum of each row in the similarity matrix, and then taking the softmax over the resulting vector. This gives us the summarized (because of the max operation) question attention distribution over context hidden states, which we denote as β. We use this Q2C attention distribution to take the weighted sum of context

hidden states and obtain Q2C attention output $\boldsymbol{c}$.

$$\mathbf{b_i} = \max_j S_{ij} \in \mathbf{R} \quad \forall i \in \{1, \ldots, N\}$$
$$\beta = \text{softmax}(\mathbf{b}) \in \mathbf{R^N}$$
$$\mathbf{c} = \sum_{i=1}^{N} \beta_i \boldsymbol{h}_i \in \mathbf{R}^{2h}$$

Finally, for each context location i, we combine context hidden state $\boldsymbol{h}_i$, C2Q attention output $\boldsymbol{a}_i$ and Q2C attention output $\boldsymbol{c}_i$ to obtain attention flow layer output $\boldsymbol{g}_i$:

$$\boldsymbol{g}_i = [\boldsymbol{h}_i; \boldsymbol{a}_i; \boldsymbol{h}_i \circ \boldsymbol{a}_i; \boldsymbol{h}_i \circ \boldsymbol{c}] \in \mathbf{R}^{8h} \quad \forall i \in \{1, \ldots, N\}$$

3.2.4 Modeling Layer

Modeling layer consists of two layers of bi-directional LSTMs with hidden size h to scan the context. Since the biLSTMs take the question aware context representation from attention flow layer, they capture the relation between context words conditioned on question. In equations:

$$\{\tilde{\boldsymbol{g}}_1, \ldots, \tilde{\boldsymbol{g}}_N\} = \text{biLSTM}\big(\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N\}\big)$$
$$\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_N\} = \text{biLSTM}\big(\{\tilde{\boldsymbol{g}}_1, \ldots, \tilde{\boldsymbol{g}}_N\}\big)$$

where $\mathbf{m}_i = [\overrightarrow{\mathbf{m}_i}; \overleftarrow{\mathbf{m}_i}] \in \mathbf{R}^{2h}$ for $i = 1, \ldots, N$.

3.2.5 Output Layer

BiDAF paper utilizes a biLSTM in the output layer for obtaining logits for the end location. We observed degraded performance with the output layer used in BiDAF paper, hence decided to use the output layer in the baseline model. More specifically, we use a fully connected layer with ReLU activation followed by two separate softmax layers to obtain start and end location distributions over the context. In order to obtain the logits before taking the softmax, we add downprojecting linear layers after the fully connected layer. In mathematical terms:

$$\tilde{\boldsymbol{m}}_i = \text{ReLU}(W_{FC}\boldsymbol{m}_i + \boldsymbol{v}_{FC}) \in \mathbf{R}^h \quad \forall i \in \{1, \ldots, N\}$$
$$\text{logits}_i^s = \boldsymbol{w}_s^T \tilde{\boldsymbol{m}}_i + v_s \in \mathbf{R} \quad \forall i \in \{1, \ldots, N\}$$
$$\text{logits}_i^e = \boldsymbol{w}_e^T \tilde{\boldsymbol{m}}_i + v_e \in \mathbf{R} \quad \forall i \in \{1, \ldots, N\}$$
$$p^s = \text{softmax}(\text{logits}^s) \in \mathbf{R}^N$$
$$p^e = \text{softmax}(\text{logits}^e) \in \mathbf{R}^N$$

3.2.6 Training Loss Function

We define our loss function as the sum of the cross-entropy losses for start and locations. More specifically, if the true start and end locations are $i_s \in \{1, \ldots, N\}$ and $i_e \in \{1, \ldots, N\}$ respectively, then the loss for a single example is

$$loss = -\log p_{i_s}^s - \log p_{i_e}^e$$

We take the average loss across the batch during training.

3.2.7 Test Time Prediction

We use the span selection strategy suggested by Chen et al. [7] during test time. In particular, we predict the span from token i to token j that maximizes $p_i^s p_j^e$ for which $i \leq j \leq i + 15$.

4 Experiments

4.1 Dataset

As previously mentioned, SQuAD reading comprehension dataset is used for the experiments. Training set contains around 86K question-answer pairs, and development set contains around 10K.

Based on the distribution of question, context, answer and word lengths in the training set, maximum lengths are all adjusted from their initial values in the baseline. Histograms for the lengths are shown in Figure 2. We set maximum length of context to 400, maximum length of question to 30, maximum length of answer to 15, and maximum character length of the words to 20. Smaller maximum lengths for each helps to use less memory, thus it speeds up the training process.

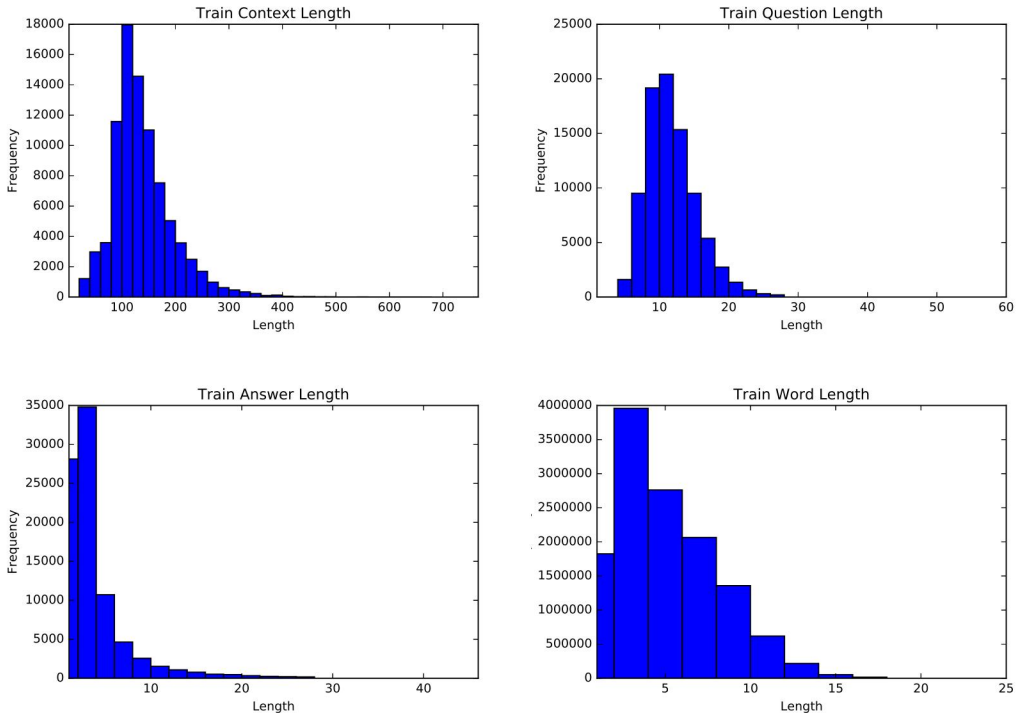

Figure 2: Histograms of context, question, answer and word lengths in the training set

4.2 Experiments

In our implementation of the network described in Figure 1, we pad the contexts and questions to their maximum lengths, which are reset to 400 and 30 respectively, with a special *padding* token. We also pad character representations of words to their maximum character length, which is reset to 20, in a similar manner. We use padding masks for logits before feeding them into the softmax layers to get attention distributions or the probability distributions. These masks make sure that padding process mathematically doesn't affect the calculations of the neural network. In order to avoid exploding gradient problems in our LSTM layers, we employ gradient clipping with a maximum gradient norm of 5. We use $d = 100$ dimensional GloVe word vectors for word embeddings, since GloVe vectors with larger dimensions significantly slow down the training process. We use a hidden state size h of 200, ADAM optimizer with a learning rate of 0.001 and a batch size of 100 during training. For the character level CNN, our character embedding size d_c is 20, convolution kernel size is 5 and the number of filters f is 100, as suggested in the handout. We use dropout with a drop probability of 0.3 [11] in all LSTM layers to reduce overfitting. However, we still experience a 15.6 point F1 score difference between the training and the dev set after 22.5K iterations. We stop training when EM and F1 scores for the dev set plateaus.

4.3 Results and Analysis

We use F1 score, the harmonic mean of precision and recall, and Exact Match (EM) score to evaluate our model. A single model of our implementation achieves 78.3% F1 and 69.2% EM on the test set. We compare our results to the current state of the art in Table 1. We have higher EM and F1 scores

Table 1: Model performance comparison to state of the art

Model	Dev Set (EM/F1)	Test Set (EM/F1)
Human Performance	- / -	82.3 / 91.2
R-NET (ensemble)	76.7 / 83.7	82.1 / 88.1
BiDAF (ensemble)	72.6 / 80.7	73.7 / 81.5
BiDAF (single)	67.7 / 77.3	68.0 / 77.3
Dynamic Coattention (ensemble)	70.3 / 79.4	71.6 / 80.4
Match-LSTM with Ans-Ptr (ensemble)	67.6 / 76.8	67.9 / 77.0
Baseline	35.1 / 44.3	34.8 / 44.2
Our Model (single)	67.8 / 77.5	69.2 / 78.3

Table 2: Performance analysis of our model and its ablations on the dev set.

	Dev Set EM	Dev Set F1
Basic C2Q attention	66.4	75.9
No char embedding	65.2	75.4
Independent start/end selection	67.0	76.5
Our Model (single)	67.8	77.5

than the single model implementation of original BiDAF, possibly due to our test time span selection strategy and our different output layer.

4.3.1 Ablation Analysis

Table 2 shows ablation analysis results of our model when the hyperparameters described in section 4.2 are used. To evaluate bi-directional attention performance, we replace it with the basic C2Q attention layer we used in the baseline. Having attention flow in both ways improves the F1 and EM scores by approximately 2.5%. Removing character embedding layer from our model results in a decrease in EM and F1 scores by 2.6% and 2.1%, respectively. It is mainly because char-level representations can handle out-of-vocabulary (OOV) words better. Finally, we change our smart test time prediction strategy to taking the argmax over start and end distributions independently, similar to the baseline. This approach decreases the performance by approximately 1%, showing the importance of smarter span selection at test time.

4.3.2 Attention and Prediction Analysis

Query-to-context (Q2C) attention is a key component in answering the query, since it shows which context words have the closest similarity to the query words. We show the Q2C attention matrix in Figure 3a. Our attention mechanism successfully captures the most critical components of the question, 'acl', 'career' and 'thomas', in the context. We also show the start and end probability distributions in Figure 3b. Our model very confidently places the start and end locations to the correct answer 'three', indicated by a single bright yellow column in the probability vector.

Context-to-query (C2Q) attention signifies which query words are the most relevant to each context word. We list the 3 most relevant context words for each query word in Figure 5. We can see that our attention mechanism works well from the following examples: 'suffered', 'broken' and 'tears' attend to the question word 'injuries' the most. Quantitative context words like 'three', 'major' and '11-year' attend the question word 'many' the most.

4.3.3 Error Analysis

We do error analysis on our model based on question type and answer length. Our model's performance decreases as the ground truth answer length increases. However, since 95% of the answers in the dev set have a span shorter than 6 words, our overall performance is mainly affected by the questions with short answers.

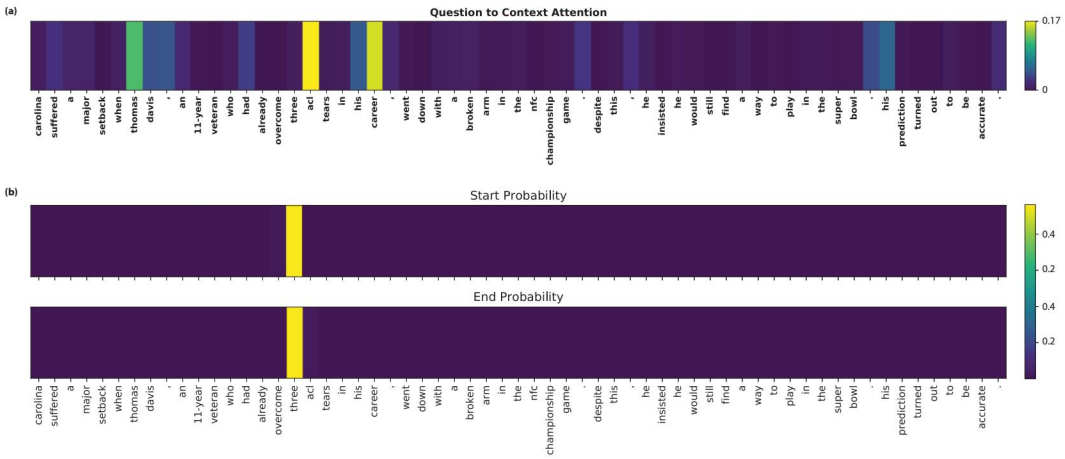

Figure 3: (a) Question to Context attention for an example context, question pair. (b) Start and end probability distributions for the same example.

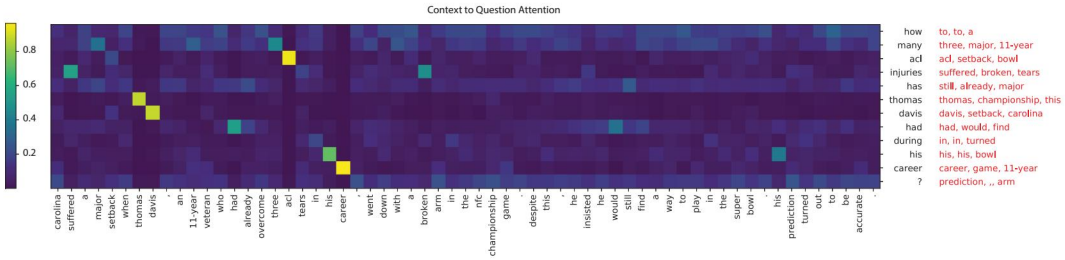

Figure 4: Context to Question attention for an example context, question pair. For each question word, top 3 context words with the highest attention are shown in red color.

We report our model's performance for different question types in Table 4. In general, the model performs better whenever the question has a shorter average answer length. The model particularly excels at "when" questions with a %83.5 EM and %87.2 F1 score.

We show the model's performance on ground truth answers of different lengths in Table 3. As we increase the length of the answer, the scores get worse. EM score is zero for answers that are longer than 15 words since we set the maximum answer length to 15.

We select some cases from the dev set where our model gives incorrect answers to analyze and consider improvements. There are several types of common mistakes that our algorithm makes. Most mistakes are due to imprecise boundaries in the prediction. In particular, the algorithm either omits the first and/or the last few words of the true answer or it gives a longer answer than it should. Although it's hard to get rid of these problems completely, it might be helpful to implement more sophisticated techniques for start and end location prediction.

Another very common problem is paying attention to the wrong region of the context, which is illustrated by the example below and its corresponding Q2C attention plot in Figure 5.

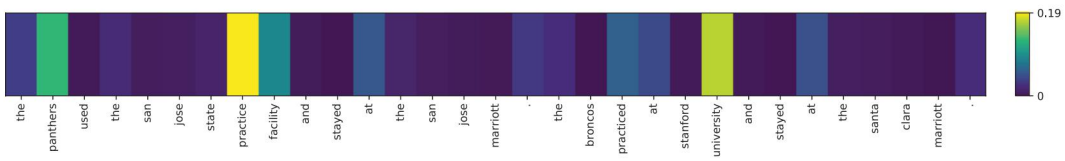

Figure 5: Question to Context attention for a question that the model gives an incorrect answer.

Table 3: Model performance based on ground truth answer length in the dev set. Since we limit our maximum answer length to 16, EM score for answer lengths greater than 16 is 0.

Answer Length	EM/F1	#Questions
1-5	69.0 / 78.0	10068
6-10	47.5 / 70.0	436
11-16	23.0 / 54.9	61
16+	0 / 56.6	5

Table 4: Model performance based on question type in the dev set

Question Type	EM / F1	Avg. Answer Length	#Questions
When	83.5 / 87.2	2.04	696
Who	77.1 / 82.8	2.43	1061
Which	68.7 / 76.6	2.42	454
How	68.3 / 78.6	2.42	1090
Where	63.7 / 76.2	2.68	433
What	62.9 / 74.0	2.96	4753
Why	43.7 / 69.1	6.31	151

Question: At what university's facility did the Panthers practice?
Context: The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott. The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott.

Although model pays significant attention to "practice" in the "San Jose practice facility", it fails to capture "facility" in the context. Since "Stanford University" has "practiced" coming before it in the context, model pays a lot of unnecessary attention to "university". In the end, it predicts "Stanford University" instead of the correct answer "San Jose State". These mistakes could be resolved by adding more layers while computing query-to-context and context-to-query attentions. Our model also makes mistakes that are much less common such as paraphrasing problems and syntactic complications.

5 Conclusion

In this paper, we implement BiDAF network that models query-aware representations of the context paragraph without early summarization. We make modifications in the modeling layer and in the span selection strategy. Our experiments demonstrate that our model achieves the state-of-the-art results in SQuAD dataset. We provide ablation analyses to show the functionality and the importance of each module in our model. Our visualizations for the attention mechanism and for the start and end probability distributions give us better insight on how our model works. Our analyses based on question types and answer lengths demonstrate in which cases our model makes mistakes. Future work could involve building a deeper network for the attention component to resolve issues with paying attention to the wrong region of the context paragraph. In addition, implementing different methods for start and end location prediction such as the one in Dynamic Chunk Reader could help increase the performance.

References

[1] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[3] Pranav Rajpurkar. The stanford question answering dataset. `https://rajpurkar.github.io/SQuAD-explorer/`. [Online; accessed 20-March-2018].

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[5] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016.

[6] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *ACL*, 2017.

[7] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017.

[8] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016.

[9] Yang Yu, Wei Zhang, Kazi Saidul Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end reading comprehension with dynamic answer chunk ranking. *CoRR*, abs/1610.09996, 2016.

[10] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.