# Question Answering System with BiDAF and DCN

**Akshay Navalakha**
akshaynavalakha@gmail.com

**Subin Modeel**
subin@apache.org

## Abstract

Machine comprehension (MC) is a challenging Natural Language Processing problem. It requires modelling the complex behaviour between the context and questions to find the answer. In this project, we have re-implemented from scratch two neural architecture for the QA task The Bidirectional Attention Flow model(BiDAF) & Dynamic Coattention Networks(DCN) with some modifications.We also experimented with using BiDAF decoder in DCN and DCN decoder in BiDAF.Our best model achieves an F1 score of 75.142% F1 and 65.593% EM on the test set with BiDAF model.

## 1 Introduction

Question answering (QA) is a task in natural language processing that requires both natural language understanding and world knowledge. In this task, a model must give answers to queries presented in natural language based on a context paragraph.The context contains the answer to the query. These problems require some attention mechanism between the context and the questions.

For the purposes of this project we use Stanford Question Answering dataset(SQUAD)[1] released by Rajpurkar et al. (2016). The SQUAD dataset is orders of magnitude larger than all previous hand-annotated datasets and has a variety of qualities that culminate in a natural QA task. SQuAD has the desirable quality that answers are spans in a reference document. This constrains answers to the space of all possible spans. However, Rajpurkar et al. (2016) show that the dataset retains a diverse set of answers and requires different forms of logical reasoning, including multi-sentence reasoning.

For the purpose of the paper we have explored a combination mainly from two neural architectures introduced in The Bidirectional Attention Flow model(BiDAF)[2] & Dynamic Coattention Networks(DCN)[3].

In addition to re-implementing a vanilla BiDAF, we have experimented with substituting the output layer of the BiDAF with the Dynamic Pointer Decoder from the DCN paper. Also, in addition a bi-directional attention a self-attention layer (inspired from RNET) was inserted to get a BiDAF + Self Attention with a dynamic decoder network. A comparison in the performance of the three versions of BiDAF is shown in table 1 .

Apart from BiDAF, a Dynamic Coattention model was re-implemented. Here we compute the affinity matrix between the context and question, which is then used to weight the continuous representations of the two documents.The decoder layer is complicated where it iteratively estimates the start and end location of the answer, allowing it to better handle local minima. Two variants of the model were implemented one with decoder mentioned in the DCN paper, second one with BiDAF Decoder.

## 2 Related Work

The SQUAD is less than 2 years old but has led to significant amount of papers in reading comprehension. Wang & Jiang (2016b) build question-aware passage representation with match-LSTM

(Wang & Jiang, 2016a), and predict answer boundaries in the passage with pointer networks (Vinyals et al., 2015). Seo et al. (2016) introduce bi-directional attention flow networks to model question-passage pairs at multiple levels of granularity. Xiong et al. (2016) propose dynamic co-attention networks which attend the question and passage simultaneously and iteratively refine answer. RNET(2018) uses gated attention between the question and context and self attention layer. These models have come very close to human performance.

## 3 Problem Statement

Given a context words $(x_1^Q, x_2^Q, ..., x_n^Q)$ denote words in the question and $(x_1^D, x_2^D, ..., x_m^D)$ we need to predict the answer span $(x_s, x_e)$ from the context which matches answer to the question. A F1 and Exact match metric is used to evaluate the performance of the model. The squad dataset is divided into 80% training dataset, 10% in dev dataset and 10% in testing set (hidden).

## 4 Dataset and Features

We use pre-trained GloVe vectors of 100 dimensional to represent the words in the context and question.Let $x_1^c, x_2^c, ..., x_n^c$ denote word embedding in the context and $x_1^q, x_2^q, ..., x_m^q$ denote the same for words in the question. We found slight significant in performance when we increase dimensional of Glove vectors but that comes at the cost of increase in training time.So we use Glove vectors of 100 dimensional to conduct more experiments

The lengths of answer, context and questions is plotted in fig 4 , fig 5 and fig 6 respectively. Based on the histogram output we decided to use a context length of 400 and question length of 30. Decreasing the context length from 600 to 400 led to a speedup of almost 2 times enabling us to run more number of experiments.

## 5 Models

In this section we describe the 2 different models and their variations. The first two models are re-implementation of the Bidaf without character level embedding[2] (referred to as vanilla BiDAF) and Dynamic Coattention network [3]. Then we interchanged the decoders of Bidaf and Co-attention to see the difference in performance resulting in what we refer to as Bidaf Dynamic.Replacing the decoder of the bidaf would provide it the flexibility of recovering local maxmima. The last two models is adding a self attention layer after the BiDAF attention layer in both vanilla Bidaf and Bidaf Dynamic.

The description of the BIDAF with Self Attention and Dynamic Pointer and Dynamic Co-attention model is as below

### 5.1 BiDAF with Self Attention and Dynamic Pointer

We did a re-implementation of the BIDAF model without the character level encoding and added/ replaced some of the layers. The addition includes using a Iterative Decoder and a self attention mechanism as shown in fig 1.

#### 5.1.1 Word Embedding Layer

Maps each word in the context and questions to a vector space using a pre-trained word em- bedding model to produce $x_1^c, x_2^c, ..., x_n^c$ words embedding for each word in context and $y_1^q, y_2^q, ..., y_m^q$ word embedding for each word in questions

#### 5.1.2 RNN Encoder Layer

:

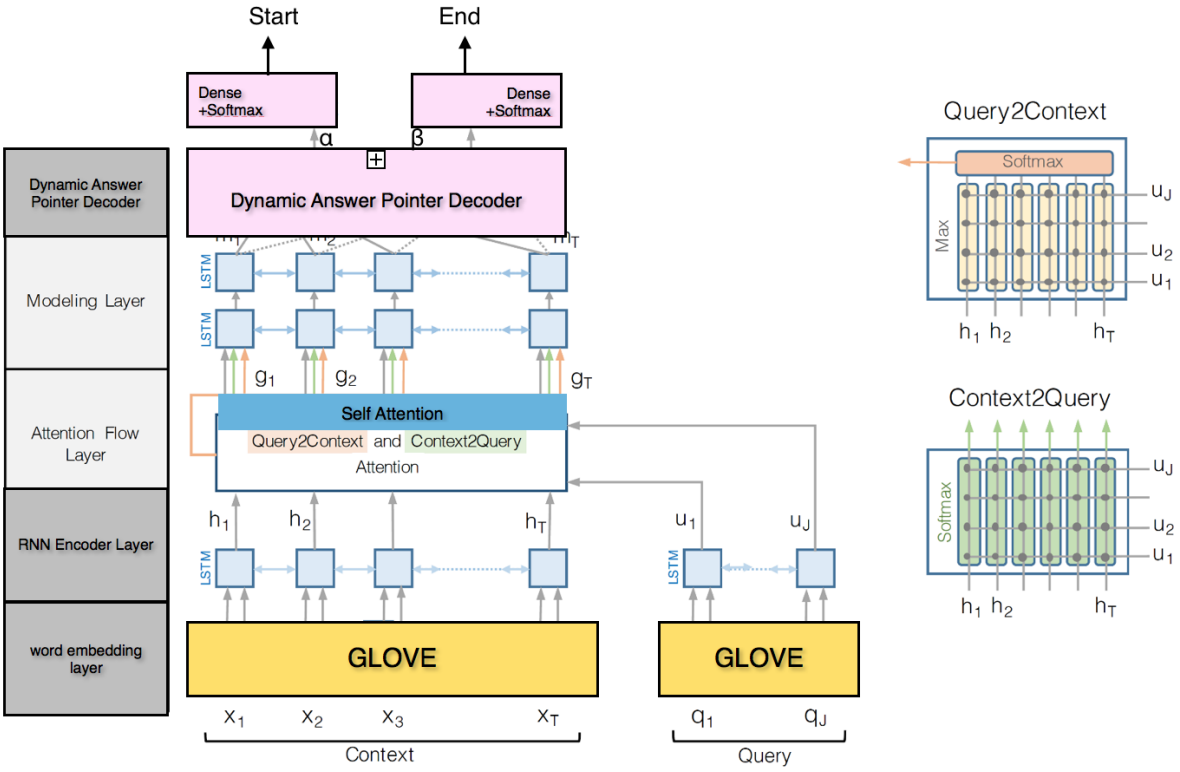


Figure 1: BiDAF with Self Attention and Dynamic Pointer Decoder.

The encoder is a bi-directional LSTM which is shared between the context and the questions. This gives us a better shared representation of the context and questions.

$$c_1, ..., c_N = BiLSTM\{x_1^c, x_2^c, ..., x_n^c\}$$

$$q_1, ..., q_N = BiLSTM\{y_1^q, y_2^q, ..., x_n^c\}$$

### 5.1.3 Bidirectional Attention

Couples the query and context vectors and produces a set of query- aware feature vectors for each word in the context. Assume we have context hidden states $c_1, ..., c_N \epsilon R^{2h}$ and question hidden states $q_1, ..., q_M \epsilon R^{2h}$. We compute the similarity matrix $S \epsilon R^{N \times M}$ , which contains a similarity score $S_{ij}$ for each pair $(c_i, q_j)$ of context and question hidden states.

$$S = w_{sim}^T [c_i; q_j; c_i o q_j] \epsilon R$$

Here, $c_i o q_j$ is an elementwise product and $w_{sim} \epsilon R^{6h}$ is a weight vector. Next, we perform Context-to-Question (C2Q) Attention. (This is similar to our baselines Attention Layer). We take the row-wise softmax of S to obtain attention distributions $\alpha^i$, which we use to take weighted sums of the question hidden states $q_j$, yielding C2Q attention outputs $a_i$. In equations, this is:

$$\alpha^i = softmax(S_i, :) \epsilon R^M \forall i \epsilon \{1, ..., N\}$$

$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j \epsilon R^{2h} \ \forall i \epsilon \{1, ..., N\}$$

Next, we perform Question-to-Context(Q2C) Attention. For each context location $i \epsilon \{1, ..., N\}$, we take the max of the corresponding row of the similarity matrix, $m_i = max_j S_{ij} \epsilon R$. Then we take the softmax over the resulting vector $m \epsilon R^N$ this gives us an attention distribution $\beta \epsilon R^N$ over

context locations. We then use $\beta$ to take a weighted sum of the context hidden states $c_i$ this is the Q2C attention output $c^{'}$. In equations:

$$m_i = max_j S_{ij} \, \epsilon \, R^N \forall i \, \epsilon \, \{1, ..., N\}$$

$$\beta = softmax(m) \, \epsilon \, R^N$$

$$c^{'} = \sum_{i=1}^{N} \beta_i c_i \, \epsilon \, R^{2h}$$

Lastly, for each context location $i \, \epsilon \, \{1, ..., N\}$ we obtain the output $b_i$ of the Bidirectional Attention Flow Layer by combining the context hidden state $c_i$, the C2Q attention output $a_i$, and

the Q2C attention output c:

$$b_i = [c_i; a_i; c_i o a_i; c_i o c^{'}] \, \epsilon \, R^{8h} \forall i \, \epsilon \, \{1, ..., N\}$$

where  represents element-wise multiplication.

#### 5.1.4 Self Attention

A self attention layer was inserted between the bi-directional layer and decoder. Even though LSTM should be able to theoretically capture the references between context of large length, practically it fails because of vanishing gradients. So a self-attention layer was introduced to relate the current word in the context with all other words.

The output of the bidirectional layer is feed into the self attention module.

$$e_j^i = tanh(W_1 b_j + W_2 bi)$$

$$\alpha^i = softmax(e^i)$$

$$a^i = \sum_{j=1}^{N} \alpha_j^i b_j$$

The output of the bidirectional layer and self attention are concatenated before sending it out to the modelling layer

$$H = [b; a]$$

#### 5.1.5 Modeling Layer

We have kept the modelling layer the same as in the original BIDAF paper. It takes in H which is $R^{Nx16d}$, which encodes the query-aware representations concatenated with the self attention. This layer captures the interaction between the context and the query. We use a bi-directional LSTM of 2 layers with a ouptut size of d for each direction.

#### 5.1.6 Dynamic Answer Pointer

The dynamic answer decoder is similar to the one used in DCN[3]. The output of the model layer is feed into the dynamic decoder. The initial state of the LSTM is all set to zero and the initial start and end word is selected to be the first word of the context. The decoder using a LSTM and a highway max network. The motivation of substituting the original decoder with the iterative reasoning decoder was that the model could recover from the local maxima. A improved performance of 1% on F1 score was achieved after substituting the original decoder with the iterative reasoning decoder. For the decoder the initial state for the start word and end word is set to the first context word. Also, the initial state of the LSTM is set to be zero valued tensor. It is expected that the model will learn from this init state and iteratively update the start and end word. The maxout layer was adapted from the tensorflow library function tf.layer.maxout and modified to work to accept batch-size as None

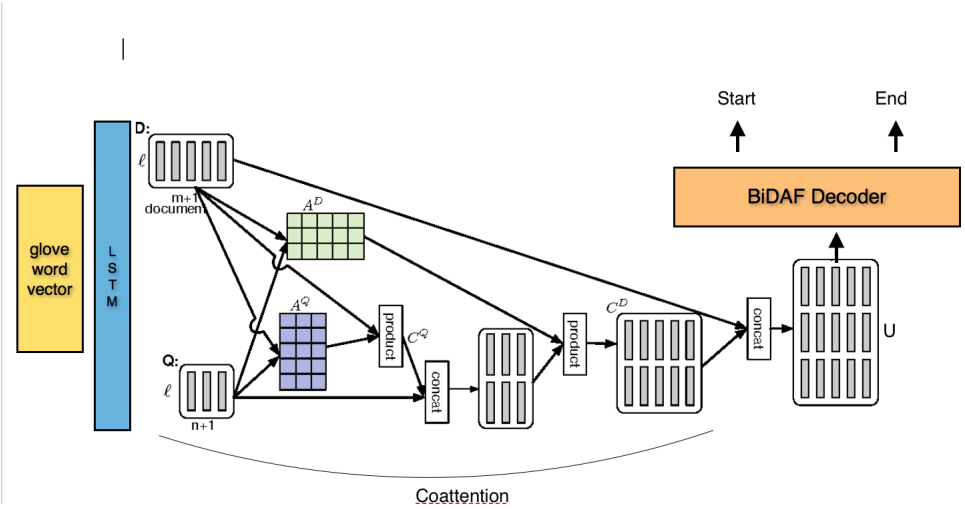


Figure 2: Coattention with BiDAF Decoder

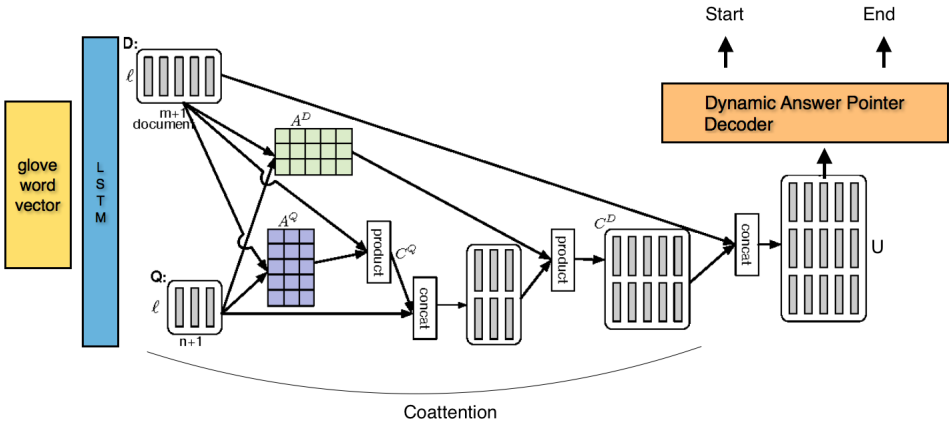


Figure 3: Coattention with Dynamic Pointer Decoder length distribution.

## 5.2 DCN

### 5.2.1 Encoder

We encode the Context as : $d_t = LSTM_{enc}\langle d_{t-1}, x_t^D\rangle$ using a LSTM. We define the context encoding matrix as $D = \langle d_1 ... d_m, d_\emptyset\rangle \ \epsilon \ R^{l\times(m+1)}$. We also add a sentinel vector $d_\emptyset$ (Merity et al., 2016), which we later show allows the model to not attend to any particular word in the input.

The question embeddings are computed with the same LSTM to share representation power: $q_t = LSTM_{enc}\langle q_{t-1}, x_t^Q\rangle$ . We define an intermediate question representation $Q = \langle q_1 ... q_n, q_\emptyset\rangle$ $\epsilon \ R^{l\times(n+1)}$. To allow for variation between the question encoding space and the document encoding space, we introduce a non-linear projection layer on top of the question encoding. The final representation for the question becomes: $Q = tanh\langle W^Q Q + b^Q\rangle \ \epsilon \ R^{l\times(n+1)}$

### 5.2.2 Coattention

The coattention mechanism attends to the question and document simultaneously, similar to (Lu et al., 2016), and finally fuses both attention contexts. Figure 2 and Figure 3 provides an illustration of the coattention models we implemented. We first compute the affinity matrix, which

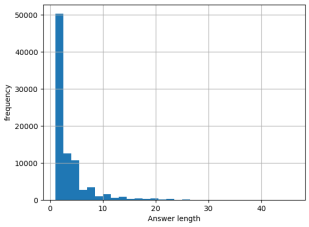


Figure 4: Answer length distribution.

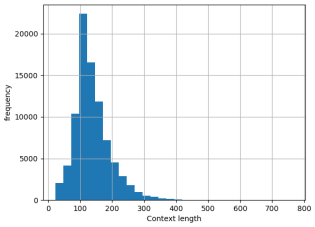


Figure 5: Context length distribution.

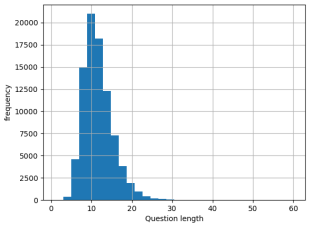


Figure 6: Question length distribution.

contains affinity scores corresponding to all pairs of document words and question words: $L = D^T Q \, \epsilon \, R^{(m+1)\times(n+1)}$. The affinity matrix is nor- malized row-wise to produce the attention weights $A^Q$ across the document for each word in the question, and column-wise to produce the attention weights $A^D$ across the question for each word in the document:

$$A^Q = softmax(L) \, \epsilon \, R^{(m+1)\times(n+1)} \; and \; A^D = softmax(L^T) \, \epsilon \, R^{(n+1)\times(m+1)}$$

Next, we compute the summaries, or attention contexts, of the document in light of each word of the question.

$$C^Q = DA^Q \, \epsilon \, R^{l\times(n+1)}$$

We similarly compute the summaries $QA^D$ of the question in light of each word of the document. Similar to Cui et al. (2016), we also compute the summaries $C^Q A^D$ of the previous attention contexts in light of each word of the document. These two operations can be done in parallel, as is shown in Eq. 3. One possible interpretation for the operation $C^Q A^D$ is the mapping of question encoding into space of document encodings.

$$C^D = [Q; C^Q]A^D \, \epsilon \, R^{2l\times(m+1)}$$

. We define $C^D$, a co-dependent representation of the question and document, as the coattention context. We use the notation $[a; b]$ for concatenating the vectors a and b horizontally. The last step is the fusion of temporal information to the coattention context via a bidirectional LSTM:

$$u_t = BiLSTM(u_{t1}, u_{t+1}, [d_t; c_t^D]) \, \epsilon \, R^{2l}.$$

We define $U = [u_1, ..., u_m] \, \epsilon \, R^{2l\times m}$ ,which provides a foundation for selecting which span may be the best possible answer, as the coattention encoding.

### 5.2.3 Decoder

We used a Dynamic Answer Pointer Decoder as mentioned in DCN and then we replaced that with a BiDAF Decoder.

## 6 Experiments

From the histogram it is seen that most of the context have length below 400. So we used context length 400 in our models. This reduced time taken to run our models. We used Adam Optimizer.We tried to build different models and compare the improvements for each of the models. We see that moving from Co-attention to Bidaf gave us 3 % points improvement on the dev leaderboard. Substituting the Bidaf decoder with the dynamic decoder gave an improved performance of 1% on F1 score with a hidden size of 100. The best model which we got was for bidaf implementation with a bidaf decoder with a hidden size of 200. We couldn't test a hidden size of 200 with the dynamic decoder because of memory constraints.

Table 1 gives a comparison of the different models performance. The best score of 75% on FM score was achieved on the test score.

Adding the self-attention layer provided provided a slight improvement in performance.

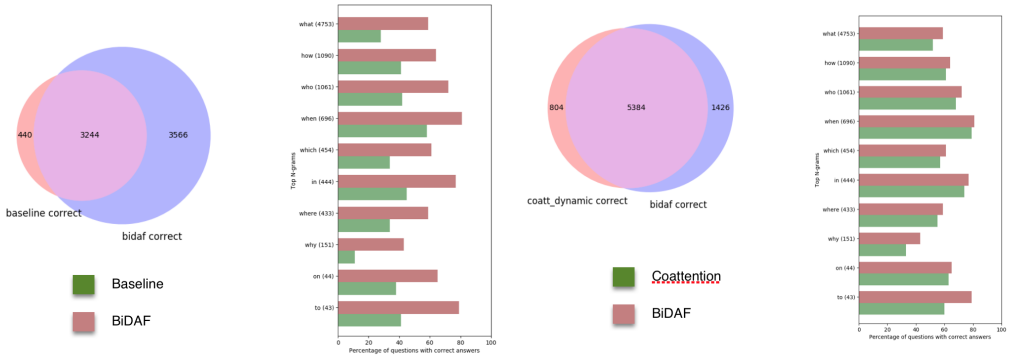


Figure 7: Venn diagram of the questions answered correctly(i.e EM) by our models(total Q=10570).& Correctly answered questions broken down by the 10 most frequent first words in the question

To reduce the over-fitting problem we tried with dropouts ranging from 0.15 to 0.3 and regularization factor of 0.0001 to 0.001. It seems that the regularization and dropout had very little effect on the model and the dev loss begins to plateau off at or near dev F1 score of 68%.

An experiment to understand the impact of learning rate was also performed. We changed the learning rate from 0.001 to 0.0001. As the learning rate decreases we get a smoother graph for the loss.

### 6.1 Analysis

In Figure 7 and 8 we can see how our model performs on different types of questions(What, how, who etc).Our model performs well on "When/Where/Who/In" questions But for reasoning questions of kind "Why" it has lower performance.All our model perform much better than the Baseline models.The Venn Diagrams show the questions answered correctly by our models.Our models performed much better than the Baseline.

Figure 9 shows our models in action. Given the context sentence "the length of the rhine is conventionally measured in rhine-kilometers rheinkilometer , a scale introduced in 1939 which runs from the old rhine bridge at constance 0 km to hoek van holland 1036.20 km . the river length is significantly shortened from the river 's natural course due to number of canalisation projects completed in the 19th and 20th century . the " total length of the rhine " , to the inclusion of lake constance and the alpine rhine is more difficult to measure objectively ; it was cited as 1,232 kilometres 766 miles by the dutch rijkswaterstaat in 2010". And the question "where does the rhine river 's measurement end ?"the probabilities for the start and end positions are shown in the figure. We have also Attached a Supplementary report[8] showing the probability distribution of start and end score for all the models we developed.

## 7 Conclusion

We were able to build a good model for answering questions presented to it in natural language. Of the 4 models we developed our BiDAF model was able to achieve better performance. As of right now, we achieve a F1 score of 75.142 and an EM score of 65.593 which is on par with some of the entries on the official SQuAD leaderboard.

### Future Work

There are a few more experiments we would like to try in the future like using more word features. The original BiDAF implementation used character encoding which was not implemented by us and there is still scope for further hyperparameter tuning and bug fixing. Coattention with self attention gave poor performace.We would like to implement DCP+[7] which is based on stacked coattention

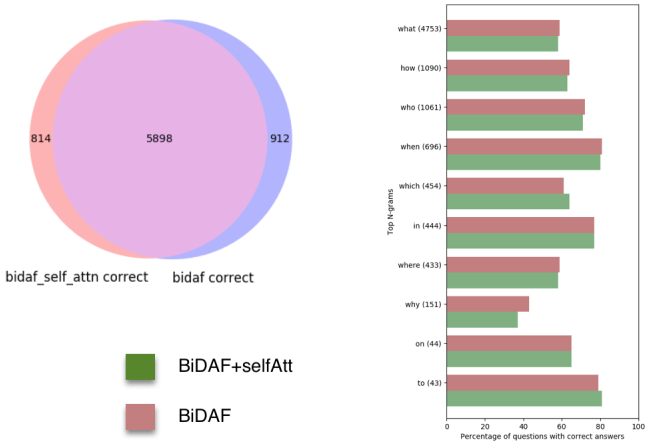


Figure 8: Venn diagram of the questions answered correctly(i.e EM) by our models(total Q=10570).& Correctly answered questions broken down by the 10 most frequent first words in the question

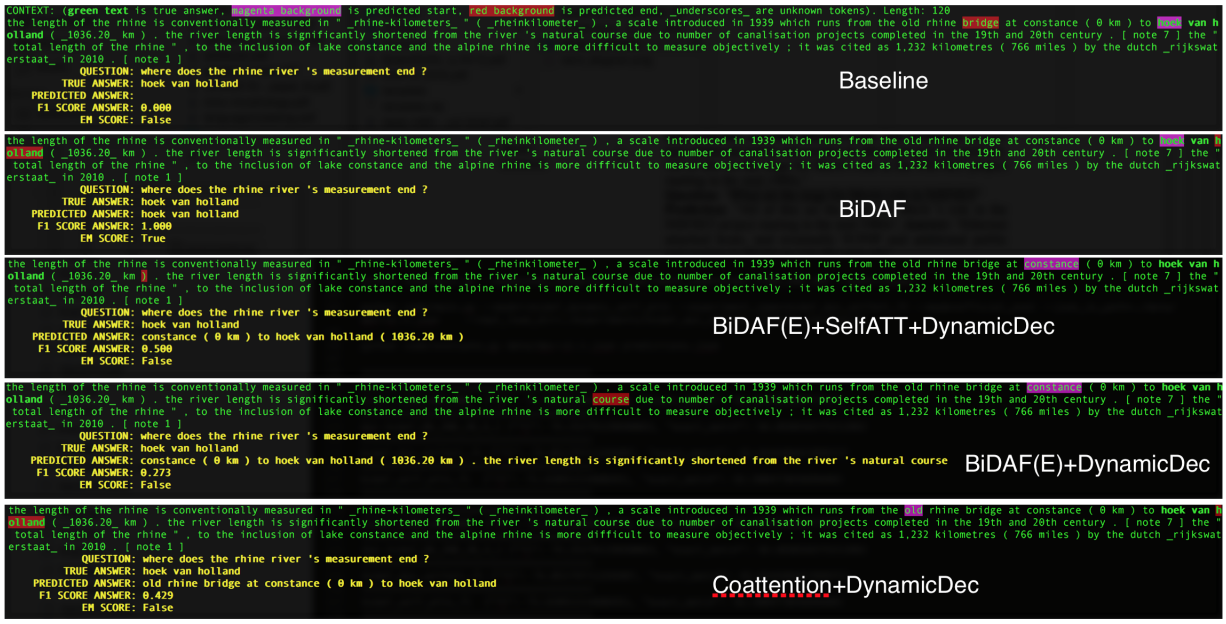


Figure 9: Our Models in Action

Table 1: Leaderboard Performance

| Model | Dev EM | Dev F1 | Test EM | Test F1 |
|---|---|---|---|---|
| BiDAF Attention + BidaF decoder (Ours) | 64.427 | 74.444 | 65.593 | 75.142 |
| BiDAF Attention + Answer Pointer decoder (Ours) | 63.652 | 74.024 | 64.597 | 74.364 |
| BiDAF Attention + Self Attention (Ours) | 63.500 | 73.638 | - | - |
| BiDAF Attention + Self Attention + Answer Pointer decoder (Ours) | 63.954 | 74.051 | - | - |
| Coattention Only (Ours) | 53.955 | 64.171 | - | - |
| Coattention with BidaF decoder (Ours) | 58.079 | 69.008 | - | - |
| Coattention with Answer Pointer decoder (Ours) | 58.543 | 69.390 | - | - |
| BiDAF | - | 76.3 | - | 76.9 |
| DCN | 65.4 | 75.6 | 66.2 | 75.9 |
| SQUAD Baseline(Ours) | 34.853 | 44.117 | - | - |
| Human (Rajpurkar et al., 2016) | 81.4 | 91.0 | 82.3 | 91.2 |

to improve performance Since Vaswani et al. (2017) show that the stacking of attention layers helps model long-range dependencies. We would also like to improve the performance of Coattention with Dynamic Pointer Decoder.

## 8 Contribution

The BIDAF model implementation, Dynamic Decoder from the Co-attention paper and Self Attention module was coded by Akshay. The encoder and attention layer from DCN for Co-attention, and a simplified BiDAF decoder for Coattention was done by Subin.Subin wrote scripts for visualizations.

## 9 Acknowledgments

We would like to thank Richard Socher, and all the TAs for this course. We would also like to give our thanks to Microsoft for providing GPUs on Azure to train our models.

## References

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, Squad: 100,000+ questions for machine comprehension of text, arXiv preprint arXiv:1606.05250, 2016.

[2] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, Bidirectional attention flow for machine comprehension, arXiv preprint arXiv:1611.01603, 2016.

[3] C. Xiong, V. Zhong, and R. Socher, Dynamic coattention networks for question answering, arXiv preprint arXiv:1611.01604, 2016.

[4] J. Pennington, R. Socher, and C. D. Manning, Glove: Global vectors for word representation., in EMNLP, vol. 14,pp. 15321543, 2014.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, 2017.

[6]Rupesh Kumar Srivastava, Klaus Greff, Jurgen Schmidhuber. Highway Networks arXiv:1505.00387 2015.

[7]Caiming Xiong, Victor Zhong, Richard Socher ,DCN+: Mixed Objective and Deep Residual Coattention for Question Answering arXiv:1711.00106.

[8]Supplementary Report document