Paying Attention to SQuAD: Exploring Bidirectional Attention Flow

Heather Blundell
Computer Science
Stanford University
hrblun@stanford.edu

Lucy Li
Symbolic Systems
Stanford University
lucy3@stanford.edu

Abstract

With the goal of automated reading comprehension, we apply a neural network with Bidirectional Attention Flow (BiDAF) to the Stanford Question Answering Dataset (SQuAD) and achieve F1 and Exact Match (EM) scores close to the original paper with a single model. We obtain a test F1 score of 76.037 and test EM score of 66.663. Our model includes Character-level CNN embeddings, a Highway Network layer, a Phrase Embedding layer, a Modeling layer, and smart span selection. We also explored expanding the model with feature engineering and an Answer Pointer output layer, which did not further improve our best model. We analyze our model's performance across categories of contexts, questions, and answers, and compare baseline attention with BiDAF.

1 Introduction

Reading comprehension for question answering is one of the many goals of natural language processing (NLP). One way to operationalize this task is to provide a passage and question pair to a machine, which selects an answer as some span in the passage. Learning to perform well on this task usually requires training on a large dataset, and the Stanford Question Answering Dataset (SQuAD) was created for this purpose [1]. Many subfields of NLP, including reading comprehension, originally depended on a heavy use of feature engineering and linguistics, but the past decade has witnessed an increasing trend towards using deep learning to tackle these problems [2]. In this project, we use the SQuAD dataset to build a neural network model for reading comprehension. We implemented several previously proposed techniques, including Bidirectional Attention Flow and Answer Pointer, on top of a basic attention baseline.

2 Related Work

Traditional statistical models used rule-based algorithms and features that combine semantic, frame, and syntactic information [3, 4, 5, 6]. A smaller predecessor dataset to SQuAD, the MCTest dataset, involves fictional stories and questions paired with multiple choice answers [3]. Another dataset for reading comprehension is the Daily Mail/CNN dataset, where some answer entity is chosen from all entities in a news article passage [6, 7].

Since the introduction of the SQuAD dataset and its public leaderboard[1], several high-performing neural network models have been used for this reading comprehension benchmark. Though the question-context-answer training input structure is the same across these models, they differ in how they relate the context and question to each other and to themselves across multiple layers. Some of these, such as Bidirectional attention flow (BiDAF), dynamic co-attention, R-Net, and fully-aware attention have utilized more complex forms of attention [8, 9, 10, 11]. Some models have incorporated a small amount of feature engineering into their deep learning model, such as the Document Reader in the DrQA model proposed by [12]. Most models are end-to-end, such as the Match-LSTM and Answer Pointer model and the Dynamic Chunk Reader [13, 14]. Other models such as ReasoNet have tried re-reading each passage multiple times, mimicking human readers [15].

These past attempts have yet to surpass human performance (91.221 F1, 82.304 EM) on SQuAD, demonstrating that reading comprehension is still a formidable problem. Our project uses the architecture of the BiDAF model as the foundation of our experiments. We compare our model's scores on the SQuAD dataset with those from previous models in Table 1.

[1]The SQuAD dataset and leaderboard: https://rajpurkar.github.io/SQuAD-explorer/

3 Dataset

The SQuAD dataset contains 107,785 question-context pairs matched with crowdsourced answers [1]. The context passages were collected from 536 Wikipedia articles, and every answer is a continuous span in a passage. These context categories cover a variety of topics, from scientific concepts such as oxygen to historical figures such as Martin Luther. The dataset also contains many types of answers, including dates, proper nouns, adjective phrases, verb phrases, and other longer constituents. Various kinds of lexical reasoning is required to find these answers, such as usage of world knowledge and synonym detection, and a model must also properly navigate syntactic variation and multi-sentence integration [1].

We computed several statistics of the data, presented in Figure 1. The mean word length over all contexts is 5.123, and the average context length is 773.144 characters or 137.711 words. Question lengths tended to be around ten words, with the average being 60.508 characters or 11.289 words. The majority of answers were extremely short, averaging 20.483 characters or 3.383 words. Answers usually appeared at the beginning of the context.

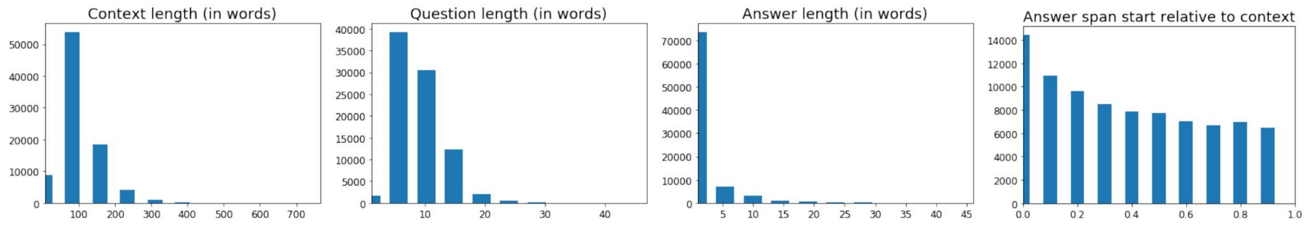

Figure 1: Histograms of context, question, and answer statistics on the SQuAD dataset. The fourth plot is based on the ratio of the answer span start index to the context length.

Using these histograms, we aimed for efficient memory usage and limited our context length to 250 tokens and answer length to 15 tokens. Only 0.41% of answer span starts were outside our maximum context length of 250. Around 2.3% of answer lengths are over 15 tokens. We wanted our model to focus on learning short answers because the bulk of the data involves those, and our baseline tended to predict incorrect answers that were extremely long (see supplementary material). The original BiDAF paper [8] does not limit the answer length, but limitation of the answer length is used by [12].

4 Approach

4.1 Best Model Architecture

Considering its high performance on the SQuAD dataset, we decided to implement a Bidirectional Attention Flow model [8]. We began with the description of BiDAF in the assignment handout and then proceeded to incorporate all of the layers in the full model, as shown in Figure 2.

First, the **Character Embedding Layer** finds the embeddings of the characters of each word in the context and question. To extract characters, we limit the vocabulary to 256 ASCII characters and the word length to 8 characters (padding or truncating when necessary). We obtain a sequence of 20-dimensional character-level word embeddings $\boldsymbol{x}_1^{char}, \ldots, \boldsymbol{x}_N^{char}$ and $\boldsymbol{x}_1^{char}, \ldots, \boldsymbol{x}_M^{char}$ for the context and question, respectively. We send these character-level embeddings through a **1D convolutional neural network** (CNN) layer with f filters and kernel size k followed by elementwise max-pooling across the word length to obtain outputs $\boldsymbol{c}_1^{char}, \ldots, \boldsymbol{c}_N^{char} \in \mathbb{R}^f$ and $\boldsymbol{q}_1^{char}, \ldots, \boldsymbol{q}_M^{char} \in \mathbb{R}^f$. We concatenate the character-level embeddings with the d-dimensional GloVe word vector embeddings to obtain the final word representations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^{d+f}$ for the context and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M \in \mathbb{R}^{d+f}$ for the question.

Next, the **Word Embedding Layer** passes the context and question representations $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ as inputs to a **highway network** [16]. Given inputs $\boldsymbol{z}$, our highway network computes $\boldsymbol{H} = \text{relu}(\boldsymbol{z}\boldsymbol{W}_H + \boldsymbol{b}_H)$ and transform gate $\boldsymbol{T} = \text{sigmoid}(\boldsymbol{z}\boldsymbol{W}_T + \boldsymbol{b}_T)$ to produce outputs $\boldsymbol{z}' = \boldsymbol{H} \circ \boldsymbol{T} + \boldsymbol{z} \circ (1 - \boldsymbol{T})$ (where $\boldsymbol{W}_H$, $\boldsymbol{W}_T$ are Xavier-initialized weight matrices and $\boldsymbol{b}_H$ and $\boldsymbol{b}_T$ are constant-initialized bias vectors). The transform gate allows for flexibility in the layer's behavior of modifying the inputs or simply passing them through. If $\boldsymbol{T}$ is close to 0 (the "carry" gate $1 - \boldsymbol{T}$ is close to 1), then the highway network can simply leave the inputs unchanged, but if $\boldsymbol{T}$ is close to 1 then the inputs are modified by $\boldsymbol{H}$. The highway network outputs matrices $\boldsymbol{X} \in \mathbb{R}^{N\times(d+f)}$ for the context and $\boldsymbol{Y} \in \mathbb{R}^{M\times(d+f)}$ for the question.

Next, the **Phrase Embedding Layer** passes $\boldsymbol{X}$ and $\boldsymbol{Y}$ through a bi-directional LSTM with hidden size h and concatenates the forward and backward outputs to obtain matrices $\boldsymbol{C} \in \mathbb{R}^{N\times 2h}$ for the context and $\boldsymbol{Q} \in \mathbb{R}^{M\times 2h}$ for the question.

Given these context hidden states $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N \in \mathbb{R}^{2h}$ (as rows of $\boldsymbol{C}$) and question hidden states $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_M \in \mathbb{R}^{2h}$ (as rows of $\boldsymbol{Q}$), the **Bidirectional Attention Flow Layer** first computes the similarity matrix $\boldsymbol{S} \in \mathbb{R}^{N\times M}$ with entries $\boldsymbol{S}_{ij} = \boldsymbol{w}_{sim}^T[\boldsymbol{c}_i; \boldsymbol{q}_j; \boldsymbol{c}_i \circ \boldsymbol{q}_j]$ where $\boldsymbol{w}_{sim} \in \mathbb{R}^{6h}$ is a weight vector (which we initialized with Xavier). Intuitively, higher entries of $\boldsymbol{S}_{ij}$ indicate pairs of context and question words that are more similar. The BiDAF model combines *Context-to-Question*

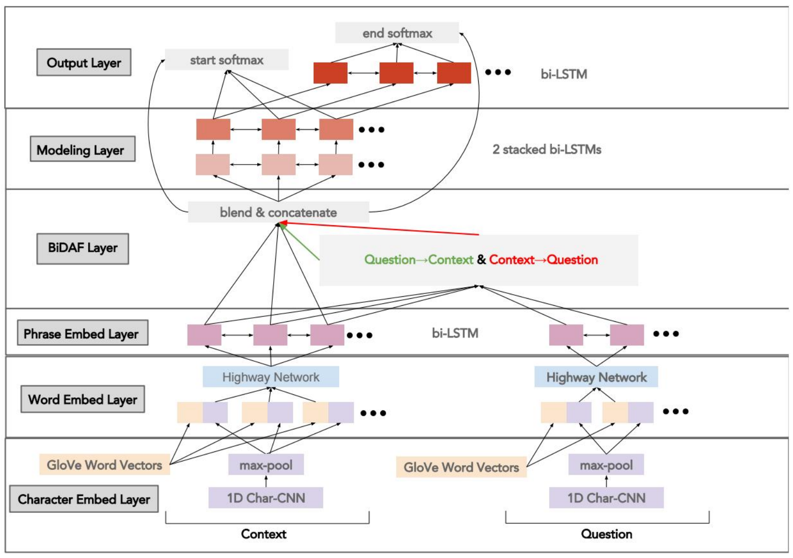

Figure 2: Model diagram. Units with the same color denote weight sharing between context and question.

(C2Q) attention and *Question-to-Context* (Q2C) attention for improved awareness of which parts of the question and context are important. C2Q Attention applies row-wise softmax to obtain attention distributions $\alpha^i = \text{softmax}(\boldsymbol{S}_{i,:}) \in \mathbb{R}^M$ for each context location $i = 1, \ldots, N$ and outputs weighted sums $\boldsymbol{a}_i = \sum_{j=1}^{M} \alpha_j^i \boldsymbol{q}_j \in \mathbb{R}^{2h}$ of the question hidden states. Q2C Attention applies row-wise max of $\boldsymbol{S}$ to obtain vector $\boldsymbol{m} \in \mathbb{R}^N$ and then obtains attention distribution $\beta = \text{softmax}(\boldsymbol{m}) \in \mathbb{R}^N$ and outputs the weighted sum $\boldsymbol{c}' = \sum_{i=1}^{N} \beta_i \boldsymbol{c}_i \in \mathbb{R}^{2h}$ of the context hidden states. Then, by combining context hidden states with the C2Q and Q2C attention outputs, we produce the query-aware *blended representations* $\boldsymbol{b}_i = [\boldsymbol{c}_i; \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{c}'] \in \mathbb{R}^{8h}$ for each context location $i = 1, \ldots, N$.

Next, the **Modeling Layer** passes the matrix $\boldsymbol{B} \in \mathbb{R}^{N \times 8h}$ (with rows $\boldsymbol{B}_{i,:} = \boldsymbol{b}_i$, $i = 1, \ldots, N$) into two stacked layers of bi-directional LSTM with hidden size h to obtain the matrix of encodings $\boldsymbol{M} \in \mathbb{R}^{N \times 2h}$. As described in the BiDAF paper [8] and verified by our experimental results, the Modeling layer is important for encoding the relationship among the context words conditioned on the question.

Lastly, in the **Output Layer**, the model outputs predicted distributions $\boldsymbol{p}^{\text{start}} = \text{softmax}(\boldsymbol{w}_{\text{start}}^T[\boldsymbol{B}; \boldsymbol{M}])$ and $\boldsymbol{p}^{\text{end}} = \text{softmax}(\boldsymbol{w}_{\text{end}}^T[\boldsymbol{B}; \boldsymbol{M}_2])$ for the each example's answer span start and end locations where $\boldsymbol{w}_{\text{start}}, \boldsymbol{w}_{\text{end}} \in \mathbb{R}^{10h}$ are weight vectors (which we initialized with Xavier) and $\boldsymbol{M}_2 \in \mathbb{R}^{N \times 2h}$ is the output of a bi-directional LSTM given $\boldsymbol{M}$ as input.

Our overall objective function is the mean of the **cross-entropy loss objective** functions for the start and end distributions for the answer span over N examples in the dataset:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} -\log \boldsymbol{p}^{\text{start}}(y_{i^{\text{start}}}) - \log \boldsymbol{p}^{\text{end}}(y_{i^{\text{end}}})$$

where y^{start} and y^{end} denote the gold start and end span locations, respectively, and θ consists of all of our trainable model parameters. We train the model in batches and use the **Adam optimizer** to minimize the loss L.

To obtain our model's **predicted** span locations l^{start} and l^{end}, we select the answer span with by maximizing the product of the probabilities of the start and end location *subject to the constraint* that the end position is at most 15 greater tokens beyond the start position. The original BiDAF paper [8] does not enforce a maximum answer length.

$$(l^{\text{start}}, l^{\text{end}}) = \arg \max_{i,\, j,\, i \le j \le i+15} \boldsymbol{p}^{\text{start}}(i) \boldsymbol{p}^{\text{end}}(j)$$

4.2 Answer Pointer

We also implemented the Answer Pointer as an alternative to our softmax output layer, as described in [13]. Answer Pointer conditions the end prediction on the start prediction. We first initialize our initial hidden state $\boldsymbol{H_0} \in \mathbb{R}^{1 \times h_q}$ using a question representation as input [10]. In the equations below, $\boldsymbol{q} \in \mathbb{R}^{h_q \times M}$ are our question hidden states, where h_q is its size returned by some previous layer, which is the Phrase Embedding Layer when we append Answer Pointer to the BiDAF model.

$$\boldsymbol{H_0} = \text{softmax}(\boldsymbol{v} \cdot \tanh(\boldsymbol{W_q q} + \boldsymbol{V_q} \otimes e_M)) \cdot \boldsymbol{q}.$$

Then, we input a context representation $\boldsymbol{c} \in \mathbb{R}^{h_c \times N}$, where h_c is its size returned by the previous layer. If the previous layer is the Modeling Layer, $c = [\boldsymbol{B}; \boldsymbol{M}; \boldsymbol{M_2}]$. We run two steps of the following:

$$\boldsymbol{s_t} = \boldsymbol{v} \cdot \tanh(\boldsymbol{W_c c} + (\boldsymbol{H_{t-1} V_c} + \boldsymbol{b_a}) \otimes e_N) \quad \text{and} \quad \boldsymbol{\beta_t} = \text{softmax}(\boldsymbol{s_t} + a \otimes e_N),$$

where $\boldsymbol{H_1} = \text{LSTM}(\boldsymbol{\beta_1} \cdot \boldsymbol{c}, \boldsymbol{H_0})$ and $\boldsymbol{s_t} \in \mathbb{R}^{1 \times N}$. The notation $a \otimes e_M$ means a is tiled M times. Learned parameters are $\boldsymbol{W_q} \in \mathbb{R}^{h_q \times h_q}$, $\boldsymbol{V_q} \in \mathbb{R}^{1 \times h_q}$, $\boldsymbol{W_c} \in \mathbb{R}^{h_q \times h_c}$, $\boldsymbol{V_c} \in \mathbb{R}^{h_q \times h_q}$, $\boldsymbol{b_a} \in \mathbb{R}^{1 \times h_q}$, $\boldsymbol{v} \in \mathbb{R}^{1 \times h_q}$, and $a \in \mathbb{R}^{1 \times 1}$. We set $\boldsymbol{p}^{\text{start}} = \boldsymbol{\beta_1}$ and $\boldsymbol{p}^{\text{end}} = \boldsymbol{\beta_2}$ as our outputs.

5 Experiments

Our experiments primarily used the Adam optimizer with learning rate 0.001 and we trained until we observed a plateau in dev F1 and EM scores. Training occurred up to $\approx$ 15K iterations for the baseline model and up to $\approx$ 10K iterations during development of our BiDAF model (approximately 7-8 hours on a single GPU).

5.1 Baseline Attention

The development of our best model described in Figure 2 and the previous section began with a baseline model, provided to us by the teaching staff of CS 224N. It contained three components: a bidirectional GRU encoder layer, a dot-product attention layer, and a fully-connected output layer with ReLU activation. We used softmax to produce our start and end answer span distributions, cross entropy loss, and an Adam optimizer. Our baseline at 15k iterations had a F1 score of 43.434 and EM score of 34.418. All further models are built upon the framework of this baseline.

5.2 DrQA Features

Our first experiment appended two lightweight features to each context word embedding c_i, as done in [12]'s DrQA model. These features were a binary indicator of whether c_i also appears in the question and the output of the word embedding for c_i attending to the question embeddings. We chose not to include part of speech, term frequency, and named entity features because they would require time-consuming preprocessing and the original paper showed they yield lesser improvement. Incorporating these DrQA features on top of our baseline and training for 6.5k iterations resulted in a F1 score of 64.582 and EM score of 53.463. However, when we later combined these features with BiDAF + Char-CNN + Span-Constraints, we observed a decrease in the model's dev performance during training. This suggests that these features may be redundant when paired with more complex forms of attention and low-level character features, as we will discuss in our analysis.

5.3 Best Model Development

5.3.1 Initial Bidirectional Attention Model

The implementation of the BiDAF model began by replacing the baseline model's attention with a bidirectional attention layer and achieved $\approx 7\%$ increase in dev F1 score and $\approx 6\%$ increase in dev EM score. Since this model is much more complex, the memory requirements are higher (especially for blended attention representations), so we reduced the maximum context length from the default 600 to a limit of 250 tokens, as mentioned in our Dataset section.

To deepen the model's interpretation of the relationship between the context and question, we incorporated bi-directional LSTM layers before and after the attention layer (Phrase and Modeling layers) and achieved a significant $\approx 23\%$ additional increase in dev F1 and EM scores. At this point, our F1 and EM scores on the dev leaderboard were 73.406 and 63.671, respectively.

5.3.2 Prediction Strategy Improvements

With a powerful model, we anticipated that additional F1 and EM score gains would become increasingly tough to achieve based on architecture alone. Hence, we investigated other strategies besides just the architecture. We realized that our model sometimes output predictions that were not valid answer spans. Also, our model sometimes predicted excessively long spans. We decided to enforce that the predicted span's end position must be greater than the start position, but at most 15 greater. As a result, our F1 and EM scores increased by $\approx 1\%$ and $\approx 0.7\%$, respectively. We also experimented with maintaining exponential moving averages of the model's trainable parameters and using these average parameters instead of the final parameters when making predictions, but did not observe improvement in the F1 and EM scores.

Also, since we have have a good estimate of the distribution of answer lengths (using our Dataset histograms), we experimented with incorporating a prior distribution for the model's predicted start and end location distributions. We encouraged the model to predict shorter answers. If the probabilities of two spans (i_1, j_1) and (i_2, j_2) were equal ($\boldsymbol{p}^{\text{start}}(i_1)\boldsymbol{p}^{\text{end}}(j_1) = \boldsymbol{p}^{\text{start}}(i_2)\boldsymbol{p}^{\text{end}}(j_2)$), then we would make the model select the span with shorter length by dividing the product of the start and end probabilities by the span length before we performed the $\arg\max$ operation. However, the effect turned out to be too strong in discouraging the model from predicting longer correct answers. As can be seen in our supplementary material document, our best model's predicted answer length distribution was already performing well without the restrictions of an answer length prior.

5.3.3 Weight Sharing Between Context and Question

Since we wanted the model to understand similarities between the context and question, we modified our BiDAF model at the Phrase Embedding Layer to use a *shared* bi-directional LSTM for the context and question at the Phrase Embedding Layer instead of two separate bi-LSTMs. This resulted in $\approx 1.1\%$ increase in F1 score and $\approx 1.2\%$ increase in EM score.

5.3.4 Character-Level CNN

To give our model additional input features and improve its generalization to unseen words, we extracted characters from the context and question, embedded them into vectors, and applied a character-level CNN. Initially, we had separate CNNs for the context and question, but deciding to share the CNN gave us a slight performance boost. Overall, the enriched feature space gives us more than 1% increase in F1 and $\approx 2\%$ increase in EM score.

5.3.5 Highway Layer

We also implemented the highway network (Word Embedding Layer) mentioned in the BiDAF paper [8]. This layer contributes a slight $\approx 0.3\%$ increase in F1 and EM scores. We hypothesize that the performance difference is subtle because this layer has the capability to output its inputs unchanged by having an active "carry" gate and only occasionally activating its transform gate. We use a single, shared highway network between the context and question instead of two stacked highway networks because we did not observe any performance increase with an additional highway.

5.4 Answer Pointer

When appended simply on to our baseline, we were able to achieve an increase of around ≈3% F1 and and ≈2% EM on the dev set by 14k iterations of training. However, when appended onto our BiDAF model, answer pointer decreased performance despite experiments with different hyperparameters. Often, any initial gains would diminish within 7k iterations. The results of adding answer pointer to our best model can be found in Table 3.

5.5 Hyperparameter Tuning

As we developed our best model, we primarily tuned hyperparameters using random search instead of grid search due to time constraints. It has been shown in literature [17] that randomly chosen trials of hyperparameters can be more efficient. Also, we incorporated L2 regularization into our loss function as a form of weight decay because we noticed that the parameter norm grows over the course of training. L2 regularization penalizes large weights and reduces overfitting. We add the term $\lambda(\frac{1}{2}\theta^T\theta)$ for every trainable model parameter θ. For additional regularization, we also applied Dropout to all of our bi-LSTMs and to the softmax output layer. We settled with a moderate 0.15 dropout probability and kept $\lambda = 1 \times 10^{-5}$ for L2 regularization as a balance between two forms of regularization.

For our optimizer, we experimented with the Adadelta optimizer used by the BiDAF paper [8] with learning rates 0.5, 1.0, and 0.1. Compared to the Adam optimizer, Adadelta appeared to exhibit much slower yet smoother growth in our model's dev F1 score for all three learning rates that we tried. We decided to keep Adam as our optimizer for efficiency concerns due to our time constraints. We also tried reducing the Adam optimizer's learning rate from 0.001 to 0.0003 and tried increasing the learning rate to 0.01 but did not observe any performance improvements. Hence, we kept a 0.001 initial learning rate. Nevertheless, since our model's dev F1 and EM scores tended to converge at 4000 iterations in most of our experiments, we incorporated an exponentially decaying learning rate that decays every 4000 iterations with a base of 0.5. A decaying learning rate helps prevent the model get even closer to a local minimum in the loss function when it is nearing convergence.

Some example effects of hyperparameter variation on our best model are shown in Table 4, and more details about our hyperparameter tuning can be found in our supplementary material.

During our experiments, it was interesting that dev set loss was not always predicative of F1 or EM scores, and some models had more gains in EM than F1 and vice versa. The performance of our best model in comparison to the state-of-the-art models is shown in Table 1 below.

6 Results and Analysis

6.1 Ablation Study

Table 3 above lists our best EM and F1 scores on the dev set for variations of our best model when trained for 10k iterations with all hyperparameters fixed. The Modeling Layer contributes significantly to performance, and removing it also resulted

	Dev EM	Dev F1	Test EM	Test F1
Logistic Regression [1]	40.0	51.0	40.4	51.0
Match-LSTM + Answer Pointer [13]	59.1	70.0	59.5	70.3
Dynamic Chunk Reader [14]	62.5	71.2	62.5	71.0
Dynamic Coattention Networks [9]	65.4	75.6	66.2	75.9
BiDAF [8]	68.0	77.3	68.0	77.3
DrQA [12]	69.5	78.8	70.0	79.0
ReasoNet [15]	–	–	70.6	79.4
R-Net [10]	72.3	80.6	72.3	80.7
FusionNet [11]	75.3	83.6	76.0	83.9
Ours	66.566	76.108	66.663	76.037

Table 1: Comparison of our model's performance with that of previous single models.

Hyperparameter	Value
`learning_rate`	0.001
`batch_size`	100
`dropout`	0.15
`hidden_size`	200
`context_len`	250
`question_len`	30
`embedding_size`	100
`char_emb_size`	20
`l2weightreg`	5×10^{-5}

Table 2: Some of the important hyperparameters of our best model.

	Dev F1	Dev EM
Best Model	**76.108**	**66.566**
No Char-CNN	74.837	64.494
No Highway Layer	75.834	65.941
No Modeling Layer	54.829	43.794
With Answer Pointer	75.891	66.064

Table 3: Ablations and an addition of our best model.

	Dev F1	Dev EM
Best Model	**76.108**	**66.566**
Hidden Size $h = 100$	75.713	65.601
Hidden Size $h = 150$	75.848	65.827
Dropout Probability 0.1	75.834	65.827
Dropout Probability 0.2	75.784	65.704
L2 Regularization $\lambda = 5 * 10^{-4}$	75.501	65.184
L2 Regularization $\lambda = 5 * 10^{-6}$	76.162	66.253

Table 4: Effect of hyperparameter variation on our best model.

in the model training in approximately 3 hours rather than 8 hours on a single GPU. The lack of improvement with the addition of the answer pointer could be that introducing this extra component is redundant, since the Modeling Layer already incorporates start information into its end prediction. It may also be that the incorporation of context inputs to the answer pointer from previous layers should involve more than simple concatenation.

6.2 Example Incorrect Predictions

After training our best model, we investigated example context-question pairs for which our model predicted an incorrect answer span. The correct span is labeled in green and the model's predicted (incorrect) span is labeled in red.

Example 1: Questions that require a model to simultaneously handle lexical and syntactical variation can be challenging. Here, our model must understand that "essential purpose of" and "necessary in" are synonymous. Since the question involves "in what ... oxygen ...?", we end up incorrectly fixating on "in medicine" near "oxygen" instead. It must also use world knowledge to know that respiration is a type of process.

Context: uptake of o 2 from the air is the essential purpose of **respiration** , so oxygen supplementation is used in **medicine** . treatment not only increases oxygen levels in the patient 's blood , but has the secondary effect of decreasing resistance to blood flow in many types of diseased lungs , easing work load on the heart . oxygen therapy is used to treat emphysema , pneumonia , some heart disorders (congestive heart failure) , some disorders that cause increased pulmonary artery pressure , and any disease that impairs the body 's ability to take up and use gaseous oxygen . Question: in what process is the uptake from oxygen necessary ?

Example 2: Another important type of question that our model has difficulty with is "Why ..." questions. Answering such questions requires reasoning beyond just pattern-matching to find the answer within the context. In the example below, the model fails to recognize the main topic of the context and instead mistakes the effect of oil withdrawal as the cause. The model tends to predict answers in close proximity to locations that match words in the question.

Context: **price controls** exacerbated the crisis in the us . the system limited the price of " old oil " (that which had already been discovered) while allowing newly discovered oil to be sold at a higher price to encourage investment . predictably , old oil was withdrawn from the market , **creating greater scarcity** . the rule also discouraged development of alternative energies . the rule had been intended to promote oil exploration . scarcity was addressed by rationing (as in many countries) . motorists faced long lines at gas stations beginning in summer 1972 and increasing by summer 1973 . Question: why was old oil withdrawn from the market ?

Example 3: Also, when the question is in the form "What NOUN ..." our model sometimes incorrectly finds the NOUN in the context and assumes that what follows it is the answer. For example, whereas to a human it's clear that "prime number" is the important adjective describing "theorem", the model considers the less-important adjective phrase following "theorem" to be the answer. The model's performance on this type of question could improve with Part-Of-Speech (POS) tag and Named Entity Recognition features in order to distinguish the most important modifiers of nouns.

Context: there are infinitely many primes , as demonstrated by euclid around 300 bc . there is no known simple formula that separates prime numbers from composite numbers . however , the distribution of primes , that is to say , the statistical behaviour of primes in the large , can be modelled . the first result in that direction is the **prime number theorem** , **proven at the end of the 19th century** , which says that the probability that a given , randomly chosen number n is prime is inversely proportional to its number of digits , or to the logarithm of n . Question: what theorem states that the probability that a number n is prime is inversely proportional to its logarithm ?

6.3 Attention Analysis

To compare how our best BiDAF model and baseline model interpret context-question pairs, we visualized their attention matrices. BiDAF's Context-to-Question attention seems to mimic the function of our additional DrQA features by highlighting synonymous or identical words (Figure 3). Hence, we infer that these extra features were no longer necessary once we implemented BiDAF, especially with the BiDAF model's enriched feature space of low-level character embeddings. Most of the attention density in BiDAF's Context-to-Question matrix corresponds to the start of the question. The Modeling Layer interprets the interactions among the context conditioned on the question keywords.

We observe that BiDAF focuses on the question keywords such as "what" along with the first few words that immediately follow the keyword, which happens to be a good summary of the question and hence what the model should look for in the context. This strategy works well for finding the answer span in many cases, such as the example in 3 below. In our supplementary materials, we have an additional visualization example using the question "there is growing interest in what indigenous group in the amazon ?", where BiDAF highlights "what indigenous group."

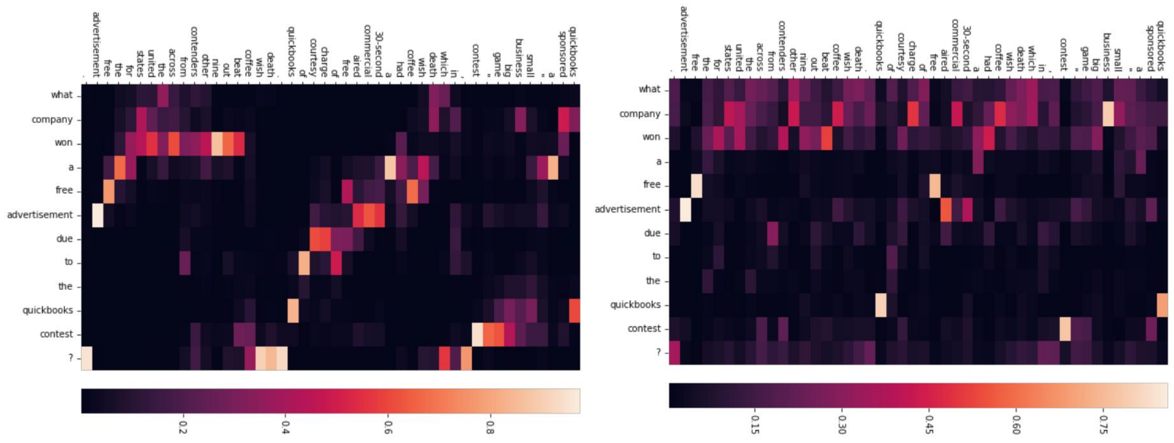

Figure 3: Context (x-axis) to question (y-axis) attention matrices for our baseline (left) and BiDAF (right).

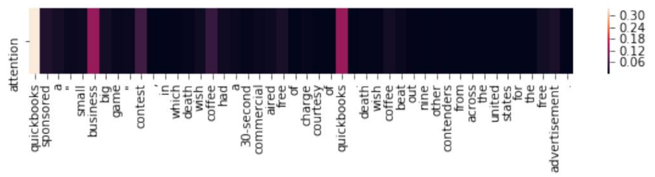

Figure 4: BiDAF question to context attention, where the context is along the x-axis. The question is "What company won a free advertisement due to the quickbooks contest?" and the correct answer is "death wish coffee."

6.4 Question, Answer, and Context Types

We split questions into types according to their starting word, and these counts and average answer lengths can be seen in Table 5. We looked at performance across different answer lengths, bucketing these lengths into ranges of 0-2, 3-4, 5-9, 10-15, and 16+ tokens. Our baseline model predicted answers that were too long, while our best model conformed to the actual length distribution (see supplementary material). Both our baseline and best model performed worse as the length of the true answer increased (Figure 7). On question types determined by starting words, our best model performed better than the baseline across all question types, but revealed the same trends. Questions starting with *when* tend to be easiest, while *why* questions are most difficult. Questions starting with *why* are also rare and have longer answers (Table 5 & Figure

word	frequency	average answer length
what	4704	3.495
who	1057	2.813
when	676	2.327
how	1045	3.045
which	451	2.818
in	436	2.165
the	233	2.627
where	431	3.443
why	150	7.233

Table 5: Words with question-starting frequency above 100.

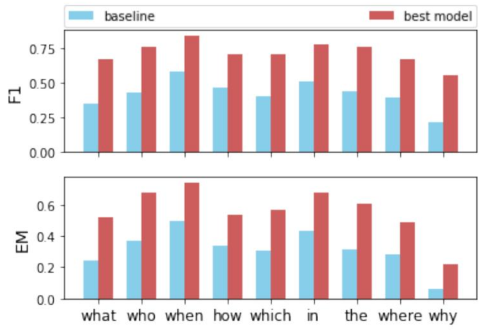

Figure 5: Scores across question types.

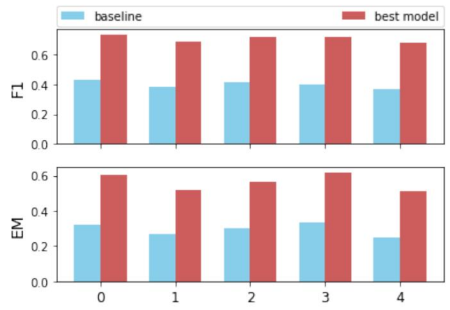

Figure 6: Score across context categories. Please see supplementary material for context article titles in each category.

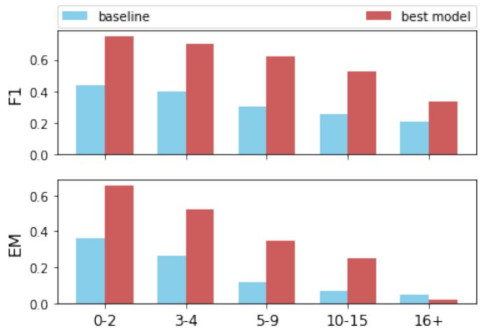

Figure 7: Score across true answer lengths.

5). If we had more *why* questions in our dataset of shorter answer lengths, we may be able to disentangle if the difficulty of these problems is due to the reasoning they involve or simply answer length. Questions that start with *how* also perform poorly though they have short answers and are less rare, which suggests that questions that require more interpretation are generally more difficult. Questions that start with *when* may be easy because their answers may fit to standard formats, such as dates, or occur around prepositions such as "during" or "after". Finally, we categorized contexts into broad topics by grouping them based on their Wikipedia article titles, creating truncated SVD tf-idf vectors for these groupings, and clustering these vectors using k-means. Our clusterings can be found in our supplementary material file, and a comparison between our baseline and best model is in Figure 6. Overall, performance for each model is consistent across topics, though it seems like historical and geographical topics (clusters 0 and 2) are easier for exact match.

7 Conclusion

Overall, by implementing a high-performing neural network for question answering on SQuAD, we learned a lot about advanced neural attention mechanisms and the process of training and tuning models. Though we obtained significant gains in F1 and EM scores by replacing the baseline attention with complex bidirectional attention, the most gains were obtained by our Modeling layer ($\approx 20\%$ increase in F1 and EM), which uses bi-LSTMs on top of the attention output to help the model interpret the context words conditioned on the question. Once we implemented our BiDAF model architecture, it became challenging to obtain further improvements beyond our official evaluation score of 76.108 dev F1. All other performance improvements were incremental: we obtained only a few percentage increase with each of our later ideas. As discussed, we spent time tuning many hyperparameters, but our selected hyperparameters appear to be locally optimal. With more time and computational resources, we would perform a thorough regularization grid search to find a more optimal combination of hyperparameters that gives our model even better generalization to held-out examples.

Besides more tuning, we have several other ideas for improving our model. We had begun implementation of R-Net [10] and fully aware attention [11], where all intermediate representations of the question and context are taken into consideration for the output. Additionally, we are interested in using Contextualized Vectors (CoVe) as inputs, which have been shown to improve SQuAD performance [18]. Since we showed that world knowledge requirements, longer answers, and *why* questions pose the biggest challenge to our model, it would be interesting to train on a large dataset focusing on these types of examples. In particular, the original SQuAD paper [1] manually categorized a sample of examples into different reasoning types, and it would be interesting to expand that annotation to the entire dataset to see how our model performs across these types [1]. After all, reading comprehension should not just involve matching the question to an answer from the context, but also *understanding* what is read.

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] C. D. Manning, "Computational linguistics and deep learning," *Computational Linguistics*, vol. 41, no. 4, pp. 701–707, 2015.

[3] M. Richardson, C. J. Burges, and E. Renshaw, "Mctest: A challenge dataset for the open-domain machine comprehension of text," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 193–203, 2013.

[4] J. Berant, V. Srikumar, P.-C. Chen, A. Vander Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning, "Modeling biological processes for reading comprehension," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1499–1510, 2014.

[5] H. Wang, M. Bansal, K. Gimpel, and D. McAllester, "Machine comprehension with syntax, frames, and semantics," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, vol. 2, pp. 700–706, 2015.

[6] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," *arXiv preprint arXiv:1606.02858*, 2016.

[7] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

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

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

[10] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, "R-net: Machine reading comprehension with self-matching networks," *Technical Report, Microsoft Research Asia*, 2017.

[11] H.-Y. Huang, C. Zhu, Y. Shen, and W. Chen, "Fusionnet: Fusing via fully-aware attention with application to machine comprehension," in *International Conference on Learning Representations*, 2018.

[12] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017.

[13] S. Wang and J. Jiang, "Machine comprehension using match-lstm and answer pointer," *arXiv preprint arXiv:1608.07905*, 2016.

[14] Y. Yu, W. Zhang, K. Hasan, M. Yu, B. Xiang, and B. Zhou, "End-to-end answer chunk extraction and ranking for reading comprehension," *arXiv preprint arXiv:1610.09996*, 2016.

[15] Y. Shen, P.-S. Huang, J. Gao, and W. Chen, "Reasonet: Learning to stop reading in machine comprehension," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1047–1055, ACM, 2017.

[16] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387v2*, 2015.

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research 13*, 2012.

[18] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Advances in Neural Information Processing Systems*, pp. 6297–6308, 2017.