Deep Question Answering on SQuAD

Mitchell Dawson
Department of Computer Science
Stanford University
Stanford, CA 94305
mjdawson@stanford.edu

Abstract

Answering questions in natural language is difficult for computer systems, which lack humans' intuitive understanding of natural language and broader knowledge of the world. We present a deep learning model for the question answering task that combines elements of the BiDAF and DrQA models. This model achieves EM and F1 scores of 62.3 and 72.4 on the SQuAD test set. We analyze the model's errors on the SQuAD development set and make suggestions for future improvement.

1 Introduction

The task of understanding a piece of text and answering questions about it in natural language is challenging-enough for humans that it is often a central component of standardized testing regimes around the world. For computer systems, which lack humans' intuitive understanding of natural language and broader knowledge of the world, the task is dramatically more difficult and has been viewed as a substantive step towards general artificial intelligence [1].

Deep learning techniques have been successfully applied to a wide range of complex, high-level problems, like image recognition and speech recognition, that had previously resisted good computational solutions [2]. These also include a number of important problems in natural language processing, like dependency parsing [3], machine translation [4], and coreference resolution [5]. Many of these impressive results natural language processing derive from the ability of deep learning techniques to learn rich vector representations of words that capture various syntactic and semantic relationships [6]. It is sensible, then, to apply these techniques to the task of question answering.

In this paper, we describe a new model architecture for the question answering problem, that combines features from two high-performing models: the Bi-Directional Attention Flow (BiDAF) network [7] and the DrQA model [8]. The paper is organized as follows. In Section 2, we describe the SQuAD dataset used to train and evaluate our model. In Section 3, we describe the BiDAF and DrQA models at a high level. In Section 4, we introduce our model and describe it in detail. In Section 5, we discuss our experiments and results and analyze the errors made by our model.

2 SQuAD Dataset

The Stanford Question Answering Dataset (SQuAD) consists of 107,785 context-question-answer tuples, drawn from 536 Wikipedia articles, which were constructed by human crowdworkers [9]. In a particular context-question-answer tuple, the answer to the question always appears word-for-word somewhere in the context. The question answering task is to, given a context and a question, output the start and end locations in the context that delineate the answer.

The SQuAD dataset is divided into training, development, and test sets on a roughly $80/10/10$ split, with the first used to train models, the second used to evaluate model performance during

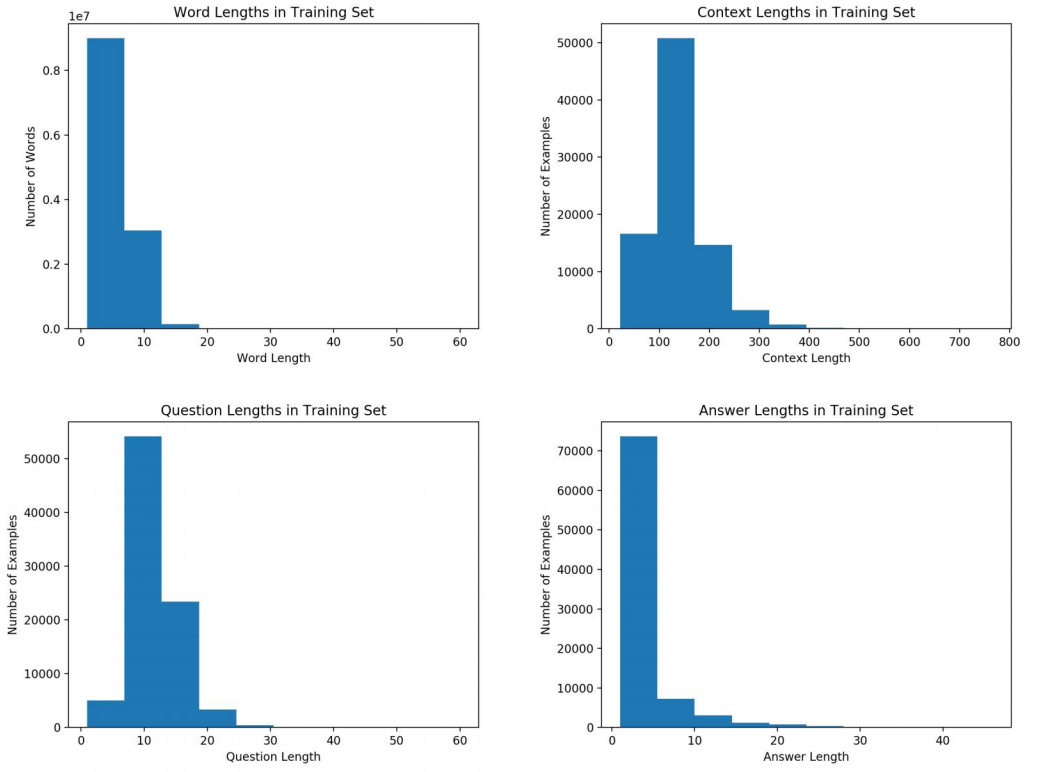

Figure 1: Training set statistics

development, and the third used to rank models on the official SQuAD leaderboard. Figure 1 shows histograms of word lengths, context lengths, question lengths, and answer lengths in the training set, and it is reasonable to believe these statistics are similar to those for the dataset as a whole. As one can see, nearly all words are shorter than 25 characters, nearly all contexts have fewer than 400 words, nearly questions have fewer than 30 words, and most answers have fewer than 15 words. This information is used to intelligently set various hyperparameters in our model.

3 Related Work

The BiDAF and DrQA models are both relatively simple models that achieved highly competitive performance on the SQuAD test set when they were first published. They share a few important elements. First, both models make use of GloVe embeddings [10] to represent the words in the input context and question. Second, they make use of multi-layer bidirectional RNNs with LSTM units to encode contextual information in both the context and the question. These encoding layers take as input word embeddings (and in the case of BiDAF, character embeddings as well) and output new representations that better capture relationships and interactions between the particular words within the given passage of text (either context or question). Third, both models make use of various kinds of attention mechanisms, with the key parts of both models involving context representations attending to question representations. In an intuitive sense, these attention mechanisms allow the models determine which words in the context are most important for answering the question.

The primary innovation in the BiDAF model was its novel attention mechanism, from which it derives its name: bidirectional attention flow. The mechanism is described in detail in section 4.5, and it offers two main advantages over simpler attention mechanisms. First, the attention is "bidirectional" in the sense that the output carries information regarding which question words are relevant to which context words and vice versa. Second, information from the inputs is allowed to "flow" through the attention mechanism since the inputs are concatenated to the attention outputs. The BiDAF also uses character-level embeddings in addition to word embeddings, which affords the

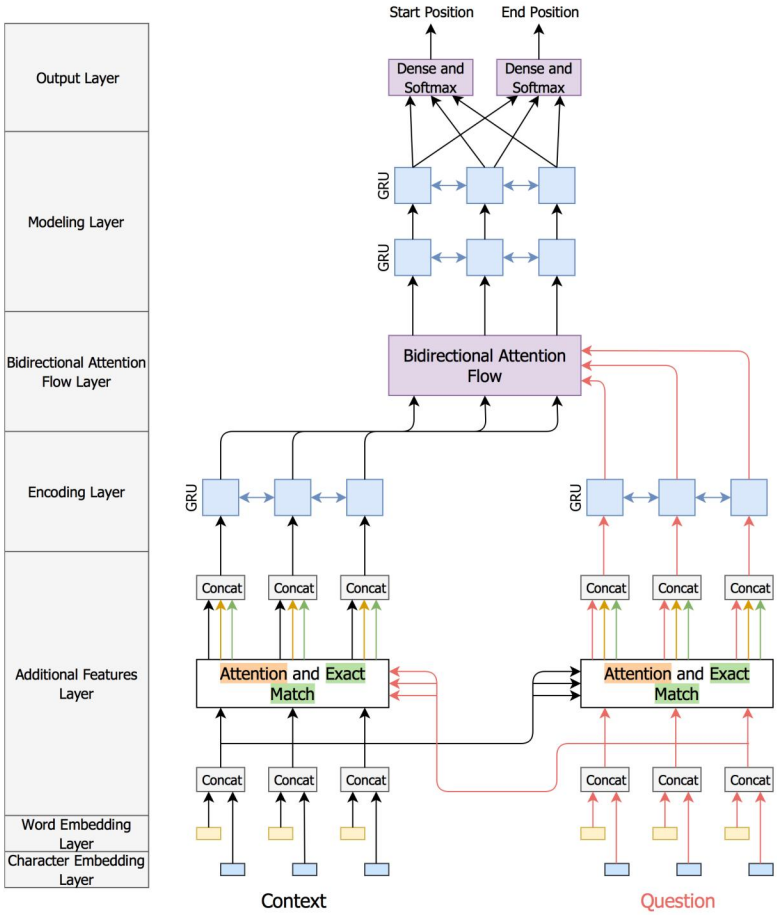

Figure 2: Model architecture

model a finer-grained understanding of the input words and also allows the model to learn something about out-of-vocabulary words.

The DrQA model, in contrast, owes its high performance to its use of constructed input features in addition to word embeddings. These features are constructed for and concatenated to the context word embeddings only and are: exact match, which is an boolean feature indicating whether or not the context word appears in the question; part-of-speech tag and named entity recognition tags; normalized term frequency; and "aligned question embedding," which is an attention mechanism where context embeddings attend to question embeddings after both are first passed through a dense layer. The authors of the DrQA model paper show that without these additional features, the performance of the rest of model, which consists of a couple of additional attention mechanisms, degrades significantly.

4 Model Architecture

Our model combines various features of the BiDAF and DrQA models and is depicted graphically in Figure 2. At a high level, the model's embedding layers, encoding layer, bidirectional attention flow layer, and modeling layer are all borrowed, with some modifications, from BiDAF. The additional features layer adds features to the embeddings that are similar to a subset of the features added in DrQA. The encoding layer uses two separately parameterized RNNs as in DrQA, and upon prediction the model restricts selected start and end positions to be within a small distance from each other, also as in DrQA. We'll now describe the full model in detail.

4.1 Inputs

The inputs to the model are: the context tokens $\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_N^t \in \{0,1\}^{|V|}$ in the form of N one-hot vectors of dimension $|V|$, where V is the vocabulary; question token $\boldsymbol{y}_1^t, \ldots, \boldsymbol{y}_M^t \in \{0,1\}^{|V|}$ of the same form; the context characters $\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_N^c \in \{0,1\}^{|C| \times W}$ in the form of one-hot matrices of dimension $|C| \times W$, where C is the character set and $\boldsymbol{x}_k^c[i,j]$ indicates whether or not the j-th character in the k-th context token is character i; and the question tokens $\boldsymbol{y}_1^c, \ldots, \boldsymbol{y}_M^c \in \{0,1\}^{|C| \times W}$ of the same form. The hyperparameters N, M, and W are respectively the lengths of contexts, questions, and words. Contexts, questions, and words shorter than these values are padded, and longer contexts, questions, and words are truncated.

4.2 Embeddings

The inputs are embedded into new representation spaces that carry more syntactic and semantic information. For token embeddings, we use GloVe embeddings, represented by the embedding matrix $\boldsymbol{E}^t \in \mathbb{R}^{d_t \times |V|}$. For character embeddings, we use our own trainable embeddings, represented by the embedding matrix $\boldsymbol{E}^c \in \mathbb{R}^{d_c \times |C|}$. Using these embedding matrices, we get token and character embeddings for the context and question: $\boldsymbol{e}_1^t, \ldots, \boldsymbol{e}_N^t \in \mathbb{R}^{d_t}$ where $\boldsymbol{e}_k^t = \boldsymbol{E}^t \boldsymbol{x}_k^t$; $\boldsymbol{e}_1^c, \ldots, \boldsymbol{e}_N^c \in \mathbb{R}^{d_c \times W}$ where $\boldsymbol{e}_k^c = \boldsymbol{E}^c \boldsymbol{x}_k^c$; $\boldsymbol{f}_1^t, \ldots, \boldsymbol{f}_M^t \in \mathbb{R}^{d_t}$ where $\boldsymbol{f}_k^t = \boldsymbol{E}^c \boldsymbol{y}_k^c$; and $\boldsymbol{f}_1^c, \ldots, \boldsymbol{f}_M^c \in \mathbb{R}^{d_c \times W}$ where $\boldsymbol{f}_k^c = \boldsymbol{E}^c \boldsymbol{y}_k^c$.

The character embeddings for both the context and question are further processed using a single convolutional layer with a 1-dimensional kernel of size k, f filters, and padding such that the output sequence is of the same length as the input sequence. Denoting this layer CNN, we get new character representations for the context and question, $\tilde{\boldsymbol{e}}_1^c, \ldots, \tilde{\boldsymbol{e}}_N^c \in \mathbb{R}^{f \times W}$ and $\tilde{\boldsymbol{f}}_1^c, \ldots, \tilde{\boldsymbol{f}}_M^c \in \mathbb{R}^{f \times W}$.

$$\tilde{\boldsymbol{e}}_k^c = \text{CNN}(\boldsymbol{e}_k^c)$$
$$\tilde{\boldsymbol{f}}_k^c = \text{CNN}(\boldsymbol{f}_k^c)$$

We then pass these outputs through a 1-dimensional max pooling layer to get our final character representations, $\bar{\boldsymbol{e}}_1^c, \ldots, \bar{\boldsymbol{e}}_N^c \in \mathbb{R}^f$ and $\bar{\boldsymbol{f}}_1^c, \ldots, \bar{\boldsymbol{f}}_M^c \in \mathbb{R}^f$. Finally, these character representations are concatenated with the token embeddings to get the overall representations of the context and question, $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N \in \mathbb{R}^{d_t+f}$ and $\boldsymbol{f}_1, \ldots, \mathbf{f}_M \in \mathbb{R}^{d_t+f}$.

$$\boldsymbol{e}_k = [\boldsymbol{e}_k^t; \bar{\boldsymbol{e}}_k^c]$$
$$\boldsymbol{f}_k = [\boldsymbol{f}_k^t; \bar{\boldsymbol{f}}_k^c]$$

4.3 Additional Features

Like the DrQA model, our model then adds additional features to the context and question representations. The first of these is an "exact match" feature, that indicates whether or not a token in the context or question appears in the other. The second is an attention feature similar to DrQA's "aligned question embedding" feature. This feature is simply the output of a simple attention layer with either the context representations attending to the question representations, or the other way round.

In particular, the exact match features are

$$\boldsymbol{m}_k^e = \mathbb{1}[\boldsymbol{e}_k \in \{\boldsymbol{f}_1, \ldots, \mathbf{f}_M\}]$$
$$\boldsymbol{m}_k^f = \mathbb{1}[\boldsymbol{f}_k \in \{\boldsymbol{e}_1, \ldots, \mathbf{e}_N\}]$$

and the attention features are

$$\boldsymbol{a}_k^e = \sum_{i=1}^{M} \text{softmax}(\boldsymbol{e}_k^T[\boldsymbol{f}_1, \ldots, \boldsymbol{f}_M])_i \boldsymbol{f}_i$$
$$\boldsymbol{a}_k^f = \sum_{i=1}^{N} \text{softmax}(\boldsymbol{f}_k^T[\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N])_i \boldsymbol{e}_i$$

We concatenate these new features to the context and questions representations to get the augmented representations $\boldsymbol{e}'_1, \ldots, \boldsymbol{e}'_N \in \mathbb{R}^{2d_t+2f+1}$ and $\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_M \in \mathbb{R}^{2d_t+2f+1}$.

$$\boldsymbol{e}'_k = [\boldsymbol{e}_k; \boldsymbol{m}_k^e; \boldsymbol{a}_k^e]$$
$$\boldsymbol{f}'_k = [\boldsymbol{f}_k; \boldsymbol{m}_k^f; \boldsymbol{a}_k^f]$$

4.4 Encoding Layer

The encoding layer consists of two, separately parameterized 1-layer bidirectional RNNs with GRU units, one for the context, biGRU^c, and one for the question, biGRU^g. We take as the context and question encodings – $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N \in \mathbb{R}^{2h}$ and $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_M \in \mathbb{R}^{2h}$ where h is the size of hidden states of the GRU units – the concatenated forward and backward hidden states produced by biGRU^c and biGRU^q, respectively, when the augmented representations are fed in as inputs.

$$\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\} = \text{biGRU}^c(\{\boldsymbol{e}'_1, \ldots, \boldsymbol{e}'_N\})$$
$$\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_M\} = \text{biGRU}^q(\{\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_M\})$$

4.5 Bidirectional Attention Flow

The bidirectional attention flow layer, a central feature of the BiDAF, first computes a "similarity matrix" $\boldsymbol{S} \in \mathbb{R}^{N \times M}$ where

$$\boldsymbol{S}_{ij} = \boldsymbol{w}^T[\boldsymbol{c}_i; \boldsymbol{q}_j; \boldsymbol{c}_i \circ \boldsymbol{q}_j]$$

Two types of attention are then computed: context-to-question and question-to-context. The context-to-question attention values, $\boldsymbol{a}_1^{c2q}, \ldots, \boldsymbol{a}_N^{c2q} \in \mathbb{R}^{2h}$ are computed as

$$\boldsymbol{a}_k^{c2q} = \sum_{j=1}^{M} \text{softmax}(\boldsymbol{S}_{i,:})_j \boldsymbol{q}_j$$

The question-to-context attention value $\boldsymbol{a}^{q2c} \in \mathbb{R}^{2h}$ is computed as

$$\boldsymbol{a}^{q2c} = \sum_{i=1}^{N} \text{softmax}(\max_{col} \boldsymbol{S})_i \boldsymbol{c}_i$$

where $\max_{col}$ denotes the column-wise maximum function. The outputs of the bidirectional attention flow layer are new context representations $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N \in \mathbb{R}^{8h}$, where

$$\boldsymbol{g}_k = [\boldsymbol{c}_k; \boldsymbol{a}_k^{c2q}; \boldsymbol{c}_k \circ \boldsymbol{a}_k^{c2q}; \boldsymbol{c}_k \circ \boldsymbol{a}^{q2c}]$$

4.6 Modeling Layer

The modeling layer consists of a single 2-layer bidirectional RNN with GRU units, biGRU^m. The outputs of the modeling layer, $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_N \in \mathbb{R}^{2h}$, are the concatenated forward and backward hidden states produced by biGRU^m when the bidirectional attention flow outpus are fed in as inputs.

$$\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_N\} = \text{biGRU}^m(\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N\})$$

4.7 Outputs and Prediction

Finally, to get the start and end probabilities, p_{start} and p_{end}, the modeling layer outputs are passed through two separately parameterized linear layers – one for each probability – that down-project the context representations into scalars, which are then passed through softmaxes.

$$p^{start} = \text{softmax}(\boldsymbol{w}_{start}^T[\boldsymbol{m}_1, \ldots, \boldsymbol{m}_N] + b_{start})$$
$$p^{end} = \text{softmax}(\boldsymbol{w}_{end}^T[\boldsymbol{m}_1, \ldots, \boldsymbol{m}_N] + b_{end})$$

The start and end positions of the answer predicted by the model are those that maximize the product of the start and end probilities and are within P of each other.

$$\text{start-pos}, \text{end-pos} = \arg \max_{i,j:0 \leq j-i \leq P} p_i^{start} p_j^{end}$$

We train the model using a cross-entropy loss function. Let $i_{start}, i_{end} \in \{1, \ldots, N\}$ denote the true start and end positions of the answer.

$$\text{loss} = -\log(p_{i_{start}}^{start}) - \log(p_{i_{end}}^{end})$$

5 Experiments and Results

Our model achieved EM and F1 scores of 62.3 and 72.4 scores on the SQuAD test set. In Table 1, this performance is compared to that of the BiDAF and DrQA models and two other benchmarks: the current top model on the SQuAD leaderboard and human performance.

We trained the model using the Adam optimizer with a learning rate of 0.001 and a batch size of 50. To reduce overfitting, we applied dropout to all RNN hidden states (in both the encoding layer and the modeling layer) and all attention outputs (both simple and bidirectional) with a drop probability of 0.25. We also added an l2 regularization term to the loss with a coefficient of $1\text{e}{-5}$. Even with these regularization efforts, the model overfit significantly, as the gap between the F1 scores on the training and development set was around 15 points. We found that increasing the regularization parameters much further reduced overall performance. We set the maximum context, question, word, and answer lengths – N, M, W, and P in the expressions above – to be 400, 30, 25, and 15, respectively. These choices were informed by the dataset statistics described in Section 2. We chose token and character embedding lengths – d_t and d_c in the expressions above – of 100 and 20, respectively; we found that increasing these lengths resulted in no significant changes in performance. We also experimented with deeper encoding and modeling RNNs and also found that these changes resulted in no significant changes in performance, while dramatically increasing the training time. We used hidden state sizes – h in the expressions above – of 200 for both the encoding and modeling RNNs. The character embedding CNN had a kernel size of 5 and 100 filters.

5.1 Error Analysis

We now examine several examples from the SQuAD development set that illustrate some weaknesses of our model.

Model	EM	F1
BiDAF	68.0	77.3
DrQA	70.0	79.0
Our Model	62.3	72.4
Best Single Model (QANet+) [11]	82.2	88.6
Human [11]	82.3	91.2

Table 1: Performance on SQuAD test set

> Context: "in july 1973, as part of its outreach programme to young people, the v&a, became the first museum in britain to present a rock concert. the v&a presented a combined concert/lecture by british progressive folk-rock back gryphon, who explored..."
>
> Question: "which musical group did the v&a present in july 1973 as part of its youth outreach programme?"
>
> True Answer: "gryphon"
> Predicted Answer: "young people"

We see in this example that the true answer is far from the portion of the context that most closely resembles the question, and that our model incorrectly predicted a phrase close to this portion of the context. This suggests our model may not be effectively modeling relationships between parts of the context that are far apart. A possible remedy for this issue may be to try using LSTM units the in the encoding and modeling RNNs instead GRU units, as LSTM units may better allow information to "flow" over long distances in the context.

> Context: "...but it was damaged when accidentally pointed into the sun. they made two evas totaling 7 hours and 45 minutes. on one, they walked to the surveyer, photographed it, and removed some parts which they returned to earth."
>
> Question: "what did the crew of apollo 12 do with parts of the surveyor they landed near after photographing them?"
>
> True Answer: "returned to earth"
> Predicted Answer: "earth"

In this example, our model's predicted answer is close to the true answer but missing the "returned to" that makes the phrase a verb phrase and therefore a sensible answer to a question about what the crew "did". This sort of error may be prevented if the model were able to model parts-of-speech information. It may be useful, therefore, to try including a part-of-speech tag feature – like that in the DrQA model – to the list of features added in the additional features layer.

> Context: "the problems with north american were severe enough in late 1965 to cause manned space flight administrator george mueller to appoint program director samuel phillips to head a 'tiger team' to investigate..."
>
> Question: "who was appointed to head a team to find the problems north america had regarding manned space flight?"
>
> True Answer: "samuel phillips"
> Predicted Answer: "george mueller to appoint program director samuel phillips"

We see in this example that our model's predicted answer spans two full names, including the correct full name. A likely source of this error is the fact that the start and end probabilities are predicted

separately. Both "george" and "samuel" likely had high start probabilities, since they're both common first names, with the start probability for "george" being slightly higher. The model therefore predicted the span from "george" to "phillips" rather than the span from "samuel" to "phillips". This issue could be addressed by predicting the entire span at once rather than predicting the start and end positions independently. This would allow the model to understand that the span from "george" to "phillips" is less likely to be the correct answer since it contains many words that aren't part of a name.

5.2 Other Experiments

In addition to implementing and experimenting with the model described in Section 4, we also implemented most of the DrQA model. There were two differences between our version of DrQA and that presented in the DrQA paper. The first was that our version lacked the part-of-speech tag, named entity recognition tag, and normalized term frequency features added by the original DrQA model. The second was that our version allowed all word embeddings to be trainable; the original DrQA model allows only the embeddings corresponding to the 1000 most frequent question words to be trainable. The performance of our model never approached that of the original, and it overfit severely. We believe this was due to the second difference and that the overfitting was caused primarily by the word embeddings over-adapting themselves to the training set.

5.3 Future Work

In the near-term we'd like to word more on systemically tuning the parameters of our model, as well as doing more to combat overfitting. In the longer-term, we'd like to experiment with the changes suggested in Section 5.1 to address some of our model's weaknesses.

6 Conclusion

In this paper, we've presented a deep learning model for the question answering task that borrows elements from the BiDAF and DrQA models. This model achieved EM and F1 scores of 62.3 and 72.4 on the SQuAD test set. A simple error analysis suggests some improvements that could address some of the model's weaknesses and boost performance. Additional work is needed tuning the model's hyperparameters and reducing overfitting. A key lesson from this project is that deep learning models can be brittle, and changing even a single part of the model (or trying to combine features from two separate models) can reduce performance if the rest is not re-tuned accordingly.

References

[1] Weston, Jason, et al. "Towards ai-complete question answering: A set of prerequisite toy tasks." arXiv preprint arXiv:1502.05698 (2015).

[2] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436.

[3] Chen, Danqi, and Christopher Manning. "A fast and accurate dependency parser using neural networks." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[4] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.

[5] Lee, Kenton, et al. "End-to-end neural coreference resolution." arXiv preprint arXiv:1707.07045 (2017).

[6] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[7] Seo, Minjoon, et al. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).

[8] Chen, Danqi, et al. "Reading wikipedia to answer open-domain questions." arXiv preprint arXiv:1704.00051 (2017).

[9] Rajpurkar, Pranav, et al. “Squad: 100,000+ questions for machine comprehension of text.” arXiv preprint arXiv:1606.05250 (2016).

[10] Pennington, Jeffrey, Richard Socher, and Christopher Manning. “Glove: Global vectors for word representation.” Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[11] “SQuAD: The Stanford Question Answering Dataset.” Leaderboard. https://rajpurkar.github.io/SQuAD-explorer/.

[12] “CS 224N Default Final Project: Question Answering.” http://web.stanford.edu/class/cs224n/default_project/default_project_v2.pdf.