BiDAF-inspired Preferential Multi-perspective Matching for Question Answering Task

Yuxing Chen
Symbolic Systems Program
Stanford University
yxchen28@stanford.edu

Kexin Yu
ICME
Stanford University
kexinyu@stanford.edu

Abstract

We combine the ideas of two high-performing SQuAD models, Bidirectional Attention Flow (BiDAF) and Bilateral Multi-Perspective Matching (BiMPM), and propose a new framework, BiDAF-inspired Preferential Multi-perspective Matching (BPMPM) for the question answering task. In particular, we consider multiple strategies when matching context and question embeddings in both Context-To-Question and Question-To-Context directions, apply a preferential rule when aggregating their results and aim for a more comprehensive view of the interaction between two sequences. We further refine the model using character-level embeddings in the Encoder Layer to better handle out-of-vocabulary words, Dynamic Pointing Decoder to recover from local maximum corresponding to initial incorrect guess in the Output layer and smarter span selection at test time.
Best Dev Score F1: 0.75071 EM: 0.64238

1 Introduction

Reading comprehension is a trending topic in natural language processing. The goal is to make machines able to "understand" the meaning of texts in general using deep learning techniques. Stanford Question Answering Dataset (SQuAD) is a new reading comprehension dataset that consists of questions posed by crowdworkers on a set of Wikipedia articles. The answer to each question is directly taken from the corresponding passage [10]. Since SQuAD was released, researchers have composed various approaches to this problem and made huge breakthroughs.

In this project, we explore and develop several deep learning systems to answer the questions in SQuAD with higher accuracies. We start with the baseline model containing a RNN encoding layer, a basic attention layer, and a output layer. A character-level CNN layer is added to obtain character-level encodings. After investigating Bidirectional Attention Flow (BiDAF) model and Bilateral Multi-Perspective Matching (BiMPM), we design a new architecture named BiDAF-inspired Preferential Multi-perspective Matching (BPMPM). We implement Dynamic Pointing Decoder to escape initial local maxima corresponding to incorrect answers [4]. The numerical measurement metrics we use are F1 and Exact Match (EM) scores.

2 Background

Attention mechanisms are becoming the best practices for sequence-to-sequence models in natural language processing and have shown remarkable success when applied to reading comprehension tasks. Attention models such as Bidirectional Attention Flow (BiDAF), Dynamic Coattention Network and Self-attention have all achieved state-of-the-art performance on the SQuAD dataset. The trend towards more complex attention is still continuing. Thus, we decide to investigate the effects of different attention types in extracting relevant aspects from sentences and revamp the attention

layer in the baseline model for SQuAD by incorporating core ideas of various attention mechanisms. Furthermore, we also seek to complement the attention based model with richer encodings from the input layer and smarter decision-making in the output layer.

3 Approach

3.1 Architecture overview

Our model consists of five layers:

1. **Character-Level Embedding Layer** generates character-level embedding by feeding each word from the context (or question) into a Convolutional Neural Network (CNN).
2. **LSTM Encoder Layer** maps each word into its pre-trained vector representation (GloVe) and concatenates it with its corresponding character-level embedding obtained from the first layer. The hybrid representation is then fed into a 1-layer bidirectional LSTM shared between the context and the question.
3. **Matching Layer** matches the context and question using four different strategies and aggregates their matching scores to produce a blended query-aware context representation.
4. **Modeling Layer** employs another Bidirectional LSTM to further encode the query-aware context representation.
5. **Output Layer** predicts the start and end position of the answer to the question.

3.2 Character Level Embedding Layer

3.2.1 Motivation

In recent works on language processing, many people apply a sequence of fixed-dimension word embeddings to represent the input text. To get the word embedding for each word, people usually use pre-trained word vectors like GloVe and Word2Vec. However, since these word vectors are trained using some specific corpora, they are likely to suffer out-of-vocabulary (OOV) problem when an unseen word appear during the test time [6]. To solve this issue, we want to extract the character-level features from each word to form a distributed representation.

3.2.2 Advantages

As mentioned in the previous section, the main advantage of using character-level encodings is that it allows us to condition on the internal structure of the words, which helps us deal with words that are not in the current vocabulary (i.e. GloVe).

3.2.3 Implementation Details

We first construct two new dictionaries `char2id` and `id2char` which help us encode the characters in the text. Suppose the dictionaries are constructed using an alphabet with size m, the character embedding size is d_c, and the word $w = [c_1, c_2, \ldots, c_L] \in \mathbf{R^L}$. Let l_0 be the parameter which represents the maximal number of characters for a word. Each word is transformed to a sequence of characters of fixed length l_0, either by padding zeros or chopping the tail. Then the character-level embedding for w is $[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{l_0}] \in \mathbf{R^{l_0 \times d_c}}$, which is the input of convolution layer of our CNN. Then we obtain a sequence of of hidden representations $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{l_0} \in \mathbf{R^{filters}}$ by applying an appropriate number of filters to each window of $[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{l_0}]$ and using ReLU activation function to introduce nonlinearities. Finally, we use max-pooling algorithm in the pooling layer to get the result $\mathrm{emb}_{\mathrm{char}}(w)$. The next step is to concatenate the character-level embedding for w, $\mathrm{emb}_{\mathrm{char}}(w)$, and the word-level embedding $\mathrm{emb}_{\mathrm{word}}(w)$ obtained from GloVe.

Based on the study of distribution of word lengths in English, we figure out that the average word length for English is 8.23, with approximately 69.1% of words have 10 or fewer letters [7]. In our model, we choose 10 as the value of l_0. The size m of our character alphabet is 70, consisting of 26 lower-case English letters, 10 digits, and 34 other characters.

The approach described above constructs a two-dimensional convolutional layer, with 4-d inputs of shape [batch_size, context_length, word_length, character_embedding_size]. We also construct an one-dimensional convolutional layer, with a flattened input of shape [batch_size, context_length*word_length, character_embedding_size], and apply the same pooling technique. The results generated by these two approaches are combined using weighted sum.

3.3 LSTM Encoder Layer

3.3.1 Highway networks

We experiment with highway networks that allow unimpeded information flow across several layers [9]. We pass the concatenation of character-level and word-level encodings to the RNN-based highway networks. Suppose for a plain feed-forward neural network, we have $\mathbf{y} = H(\mathbf{x}, \mathbf{W_H})$, where $\mathbf{x}$ is the input and H is the nonlinearity. To build a highway network, we add a transform gate T and a carry gate C. Here T and C are both nonlinear transformations:

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot C(\mathbf{x}, \mathbf{W_C}) \tag{1}$$

where T and C express how much of the output is produced by transforming the input and carrying it, respectively [9]. We simplify the equation by setting $C = T - 1$, which gives:

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \mathbf{W_T})) \tag{2}$$

3.3.2 Trainable word embeddings

The motivation is that, for different corpora, the trained word vectors may vary. When we train the model using word embeddings obtained from fixed word vectors trained on another corpus, the performance might get worse depending on how much the two corpora are different from each other. Since the dataset used in our project is not the same as GloVe, it is reasonable to make the word embeddings trainable. The word embedding variable is initialized with the pre-trained word vectors loaded from GloVe. After a few iterations, we can observe that if we plot the original word vectors and the retrained word vectors in the same space, the position of each word vector has been changed. However, these changes are not very obvious, and the relationships among these word vectors are almost preserved, as shown in Appendix A (figure 6).

Finally, each context is represented by a sequence of $(d + d_c)$-dimensional word embeddings x_1, ..., x_N and each question by a sequence of $(d + d_c)$-dimensional word embeddings y_1, ..., y_M. We then feed the embeddings into a 1-layer bidirectional LSTM (shared between the context and question):

$$\{\overrightarrow{c_1}, \overleftarrow{c_1}, ..., \overrightarrow{c_N}, \overleftarrow{c_N}\} = \text{biLSTM}(\{x_1, ..., x_N\}) \tag{3}$$

$$\{\overrightarrow{q_1}, \overleftarrow{q_1}, ..., \overrightarrow{q_N}, \overleftarrow{q_N}\} = \text{biLSTM}(\{y_1, ..., y_N\}) \tag{4}$$

3.4 Matching Layer: BiDAF-inspired Preferential Multi-perspective Matching

3.4.1 Multi-strategic Multi-perspective Matching

The matching layer is the key component of our model and is inspired by two attention mechanisms, Bidirectional Attention Flow (BiDAF) [2] and Bilateral Multi-perspective Matching (BiMPM) [3]. **First**, We define a multi-perspective matching operation as follows:

$$\mathbf{m} = f_m(c, q; W) \in \mathbf{R^l}, c, q \in \mathbf{R^h}, W \in \mathbf{R^{l \times h}} \tag{5}$$

where l denotes the number of perspectives of a particular matching strategy. Each element m_k of $\mathbf{m}$ is the matching score from the $k - th$ perspective of this strategy and calculated by the cosine similarity of the two weighted vectors:

$$m_k = cosine(W_k \circ c, W_k \circ q) \tag{6}$$

where $\circ$ is element-wise multiplication and W_k is the k-th row (or perspective) of W.

Second, We use four different matching strategies just as the BiMPM model does shown in 1:

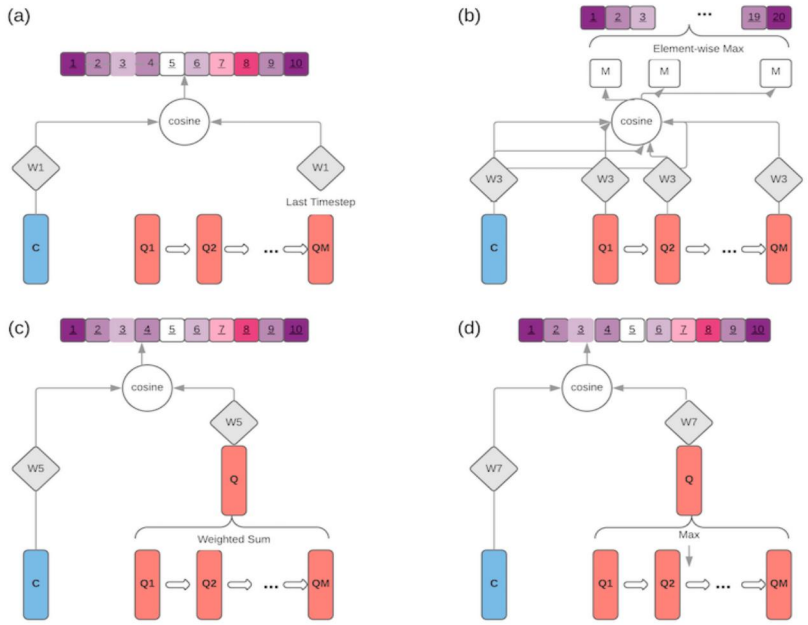

Figure 1: Diagram for strategies used in Preferential Multi-perspective Matching: (a) full matching (b) max-pooling matching (c) attentive matching (d) max-attentive matching. The max-pooling one has the largest number of perspectives.

Full matching: Each forward (or backward) context hidden state $\overrightarrow{c_i}$ (or $\overleftarrow{c_i}$) is matched against the last time step of the forward (or backward) question hidden state:

$$\overrightarrow{m_i^{full}} = f_m(\overrightarrow{c_i}, \overrightarrow{q_M}; W^1) \in \mathbf{R^l}, W^1 \in \mathbf{R^{l \times h}} \tag{7}$$

$$\overleftarrow{m_i^{full}} = f_m(\overleftarrow{c_i}, \overleftarrow{q_1}; W^2) \in \mathbf{R^l}, W^2 \in \mathbf{R^{l \times h}} \tag{8}$$

Max-pooling matching: Each forward (or backward) context hidden state $\overrightarrow{c_i}$ (or $\overleftarrow{c_i}$) is matched against each forward (or backward) question hidden state $\overrightarrow{q_j}$ (or $\overleftarrow{q_j}$):

$$\overrightarrow{m_i^{max}} = \max_{j=1,\ldots,M} f_m(\overrightarrow{c_i}, \overrightarrow{q_j}; W^3) \in \mathbf{R^l}, W^3 \in \mathbf{R^{2l \times h}} \tag{9}$$

$$\overleftarrow{m_i^{max}} = \max_{j=1,\ldots,M} f_m(\overleftarrow{c_i}, \overleftarrow{q_j}; W^4) \in \mathbf{R^l}, W^4 \in \mathbf{R^{2l \times h}} \tag{10}$$

Note we assign more weight to this matching strategy by doubling the number of its perspectives.

Attentive matching: First, calculate cosine similarities between each forward (or backward) context hidden state $\overrightarrow{c_i}$ (or $\overleftarrow{c_i}$) and each forward (or backward) question hidden state $\overrightarrow{q_j}$ (or $\overleftarrow{q_j}$)

$$\overrightarrow{\alpha_{i,j}} = cosine(\overrightarrow{c_i}, \overrightarrow{q_j}), j = 1, ..., M \tag{11}$$

$$\overleftarrow{\alpha_{i,j}} = cosine(\overleftarrow{c_i}, \overleftarrow{q_j}), j = 1, ..., M \tag{12}$$

Second, take $\overleftarrow{\alpha_{i,j}}$ (or $\overrightarrow{\alpha_{i,j}}$) as the weight of $\overrightarrow{q_j}$ (or $\overleftarrow{q_j}$) and calculate the attentive matching vector for the question sequence by taking the weighted sum of all question hidden states:

$$\overrightarrow{q_i^{mean}} = \frac{\sum_{j=1}^{M} \overrightarrow{\alpha_{i,j}} \cdot \overrightarrow{q_j}}{\sum_{j=1}^{M} \overrightarrow{\alpha_{i,j}}} \tag{13}$$

$$\overleftarrow{q_i^{mean}} = \frac{\sum_{j=1}^{M} \overleftarrow{\alpha_{i,j}} \cdot \overleftarrow{q_j}}{\sum_{j=1}^{M} \overleftarrow{\alpha_{i,j}}} \tag{14}$$

Lastly, each forward (or backward) context hidden state $\overrightarrow{c_i}$ (or $\overleftarrow{c_i}$) is matched against the corresponding attentive matching vector:

$$\overrightarrow{m_i^{attn}} = f_m(\overrightarrow{c_i}, \overrightarrow{q_i^{mean}}; W^5) \in \mathbf{R^l}, W^5 \in \mathbf{R^{l \times h}} \tag{15}$$

$$\overleftarrow{m_i^{attn}} = f_m(\overleftarrow{c_i}, \overleftarrow{q_i^{mean}}; W^6) \in \mathbf{R^l}, W^6 \in \mathbf{R^{l \times h}} \tag{16}$$

Max-attentive matching: Instead of weighed summing all question hidden states, this strategy selects the state with the largest cosine similarity as the attentive matching vector:

$$\overrightarrow{m_i^{max-attn}} = f_m(\overrightarrow{c_i}, \overrightarrow{q_i^{max}}; W^7) \in \mathbf{R^l}, W^7 \in \mathbf{R^{l \times h}} \tag{17}$$

$$\overleftarrow{m_i^{max-attn}} = f_m(\overleftarrow{c_i}, \overleftarrow{q_i^{max}}; W^8) \in \mathbf{R^l}, W^8 \in \mathbf{R^{l \times h}} \tag{18}$$

Next, the forward/backward matching scores from all perspectives are concatenated to obtain the question-aware blended representation a_i for the **Context-to-Question(C2Q) Matching**:

$$a_i = [\overrightarrow{m_i^{full}}; \overrightarrow{m_i^{max}}; \overrightarrow{m_i^{attn}}; \overrightarrow{m_i^{max-attn}}; \overleftarrow{m_i^{full}}; \overleftarrow{m_i^{max}}; \overleftarrow{m_i^{attn}}; \overleftarrow{m_i^{max-attn}}] \in \mathbf{R^{2(l+2l+l+l)}} = \mathbf{R^{10l}} \tag{19}$$

Then, to mimic the bidirectional attention flow in the BiDAF model, repeat the same process for the reverse direction and obtain the context-aware blended representation b_j for the **Question-to-Context(Q2C) Matching**:

$$b = \max_{j=1,\ldots,M} b_j \in \mathbf{R^{10l}} \tag{20}$$

where $\max_{j=1,\ldots,M}$ is the element-wise maximum.

Finally, we obtain the output c_i of the Matching Layer by blending the **C2Q Matching** vector and **Q2C Matching** vector:

$$c_i = a_i \circ b \in \mathbf{R^{10l}} \tag{21}$$

where $\circ$ is the element-wise multiplication.

3.5 Modeling Layer

We feed the matching representation into three layers of bi-directional LSTM and the result is then passed onto the output layer for prediction.

3.6 Output Layer: Dynamic Pointing Decoder with Smarter Span Selection

Dynamic Pointing Decoder (DPD): Since there might exist several candidate spans in the context, it's likely the model gets stuck in some initial local maximum [4]. With an iterative procedure which takes turns in predicting the start position and end position, our model is able to escape that initial incorrect guess.

Smarter Span Selection: We choose the start and end location pair (i, j) with $i <= j <= i + 15$ that maximizes $p^{start}(i), p^{end}(j)$, so that the predicted end position will be more reasonable and never occur before the start location [5].

4 Experiments

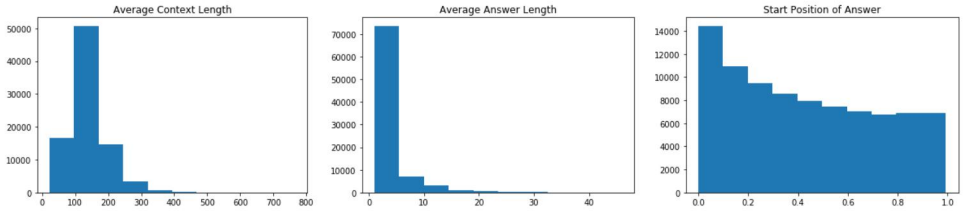

Figure 2: Distribution of context length, question length and answer position of the training data.

After exploring the distribution of context length and question length of the training data shown in 2, we set `context_len = 300` and `question_len = 30` to be the cutoff when truncating long contexts and questions. This helps decrease the model size for better runtime without losing too much information from the original sequences. Since the answer is more likely to appear at the beginning of the context, the truncation occur more often from the end (with a probability of $\frac{2}{3}$).

We've also experimented with different hyperparameter choices and end up with the following: `learning_rate = 0.001`, `dropout = 0.15`, `embedding_size` (size of the pre-trained GloVe word vectors) = `300`. We use `AdamOptimizer` as the optimization algorithm.

4.1 Evaluation Metrics

We use **F1** and **Exact Match (EM)** scores to evaluate the prediction accuracy of the improved model versions. Besides, we also care about **time of convergence** and **seconds per batch** (with 100 examples) to pursue fast learning. **Total number of parameters** is also key to the low-storage goal.

4.2 Results

To investigate the effect of each new component, we add each improvement incrementally to see how much it contributes to the overall performance. In the end, we combine these components into a final model, and build an ensemble of three with different hyperparameters to seek more robustness.

Table 1: Performance Comparison.

Model	Best F1	Best EM	Time of Convergence	Seconds per Batch
Baseline	0.39458	0.32963	15000 iterations	1.3s
BiDAF + DPD	0.57567	0.43047	9000 iterations	1.5s
BiDAF + CNN + DPD	**0.73753**	**0.63217**	4000 iterations	1.7s
BiMPM + CNN +DPD	0.51079	0.37937	5000 iterations	2.2s
BPMPM + CNN + DPD	0.58778	0.44308	4000 iterations	4.7s
Ensemble	**0.75071**	**0.64238**	-	-

4.2.1 Performance boost from character-level embeddings

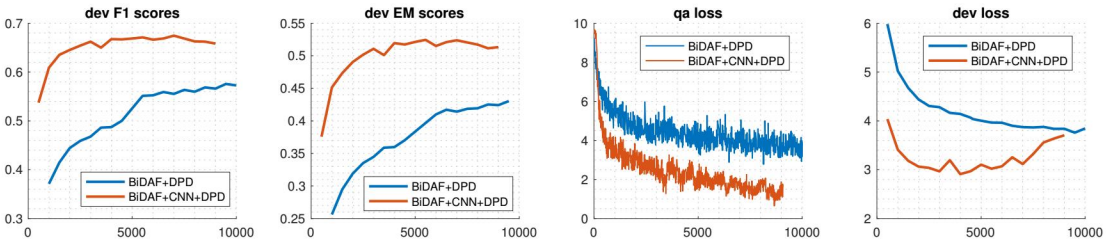

Figure 3: Performance boost from character-level embeddings

Figure 3 shows that with character-level encoding, the model achieves higher F1 and EM scores with much fewer iterations and the time needed to process each batch increases minimally.

4.2.2 Performance boost from BiDAF-inspired Preferential variant of BiMPM

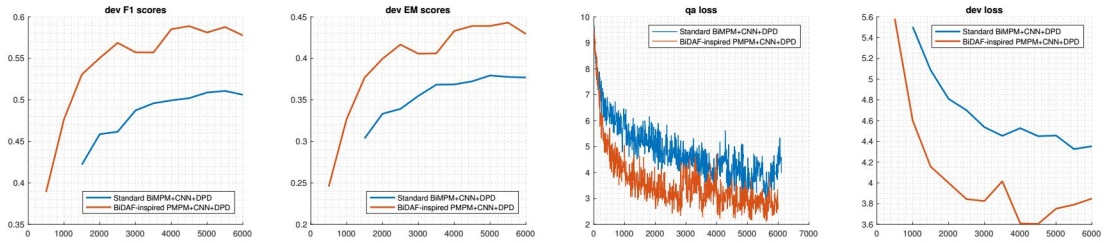

Figure 4: Performance boost from BiDAF-inspired Preferential variant of BiMPM

Figure 4 shows that applying the matching operation in both Context-to-Question and Question-to-Context directions, along with a weighting rule that prefers the max-pooling over other strategies, also contributes towards the model performance at the expense of slightly longer batch time.

4.2.3 Comparison between BiDAF and BiDAF-inspired Preferential variant of BiMPM

Here's one example where BiDAF-inspired Preferential variant of BiMPM outperforms BiDAF:

CONTEXT: in early 2012, nfl commissioner roger goodell stated that the league planned to make the 50th super bowl ``spectacular'' and that it would be ``an important game for us as a league''..
QUESTION: what *one word* did the nfl commissioner use to describe what super bowl 50 was intended to be?
TRUE ANSWER: spectacular
PREDICTED ANSWER from BiDAF: an important game for us as a league
PREDICTED ANSWER from BiMPM: spectacular

Although both models successfully locate the NFL commissioner's description of Super Bowl 50 to answer the question, only the BiDAF-inspired Preferential variant of BiMPM identifies the important requirement from the question –"one word"– that qualifies the answer. Since the BiMPM employs multiple functions when calculating the attention distribution, it gains a more comprehensive understanding of the question.

However, we also notice that the standard BiDAF model is a much more efficient learner. The loss decreases steadily from the beginning and it achieves a dev F1 score of 0.55 just with 500 iterations.

4.2.4 Visualization of Multi-Strategic Multi-perspective Matching

To understand how different matching strategies work, we first implement the standard BiMPM model and visualize their matching scores in heat map. 5 shows an example where the matching score has the largest absolute value at true/predicted answer start/end with the Full Matching strategy. Moreover, each perspective provides different advice for the matching.
To further investigate which strategy makes the best decision, we compare the scores distribution of all strategies. 5 reveals that the Max-pooling Matching strategy is the most confident and reliable. Thus, we assign higher weight to this strategy in our preferential variant of BiMPM.

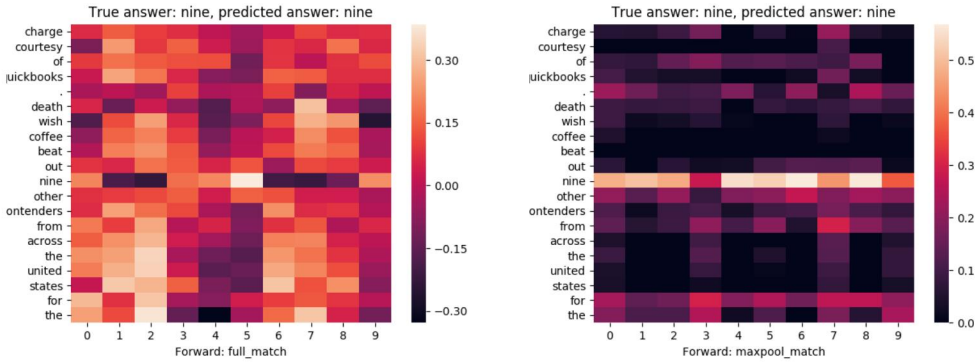

Figure 5: Full matching and Max-pooling matching scores for forward direction

5 Error Analysis

Some interesting mistakes that our model has made reveal its weaknesses and suggest how it can be further improved in the future:

CONTEXT: during the period in which the negotiations were being conducted, tesla said that efforts had been made to steal the invention. his room had been entered and his papers had been scrutinized, but the thieves, or spies, left empty-handed. ...
QUESTION: according to tesla what had been gone over by the thieves, or spies who entered his room?
TRUE ANSWER: his papers
PREDICTED ANSWER: empty-handed
Explanation: The model predicts the answer "empty-handed" because it pays attention to "the thieves, or spies". It fails to recognize that "gone over" is actually the synonym of "scrutinize" and that the question is looking for information about the object of the action rather than the subject.
Improvement: Use additional features such as the voice of the sentence (active or passive). Use fine-tuned word vector representations that capture semantic similarity.

CONTEXT: ... the english channel, the irish channel and most of the north sea were dry land, mainly because sea level was approximately 120 m (390 ft) lower than today.
QUESTION: besides the north sea and the irish channel, what else was lowered in the last cold phase?
TRUE ANSWER: english channel
PREDICTED ANSWER: dry land
Explanation: The model predicts the answer "dry land" as it looks for adjacent or parallel words for "north sea" and "irish channel". However, it fails to tell the difference between nominative case and accusative case. In fact, the true answer "english channel" corresponds to the nominative case and occurs much earlier in the sentence.
Improvement: Use additional features such as the Part-Of-Speech tag and the casing of words.

6 Conclusion

In this project, we explore the effects of different attention mechanisms, Bidirectional Attention Flow (BiDAF) and Bilateral Multi-Perspective Matching (BiMPM) in particular, on improving reading comprehension model. We compare their strengths and develop a variant by leveraging the advantages of both frameworks. All models use a CNN layer to find the character-level encodings for words in the texts. The best F1 and EM scores given by a single model (BiDAF+CNN+DPD) are 0.73753 and 0.63217, respectively. The ensemble of these models improves F1 score by around 0.013, which reaches 0.75071. As mentioned in the Error Analysis section, future work involves using additional features and fine-tuned word vector representations. We may also run more experiments to find better hyper-parameters for our models.

7 References

[1] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. *arXiv preprint arXiv:1509.01626, 2015.*

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603, 2016.*

[3] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814, 2017.*

[4] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604, 2016.*

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051, 2017.*

[6] David Golub and Xiaodong He. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727, 2016.*

[7] Parikh, Ravi. "Distribution of Word Lengths in Various Languages." Distribution of Word Lengths in Various Languages - Ravi Parikh's Website. Accessed March 18, 2018. http://www.ravi.io/language-word-lengths.

[8] Kim, Yoon, Yacine Jernite, David Sontag, and Alexander M. Rush. "Character-Aware Neural Language Models." In *AAAI*, pp. 2741-2749. 2016.

[9] Srivastava, Rupesh Kumar, Klaus Greff, and Jrgen Schmidhuber. "Highway networks." *arXiv preprint arXiv:1505.00387, 2015.*

[10] Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang. "Squad: 100,000+ questions for machine comprehension of text." *arXiv preprint arXiv:1606.05250, 2016.*

Appendix A

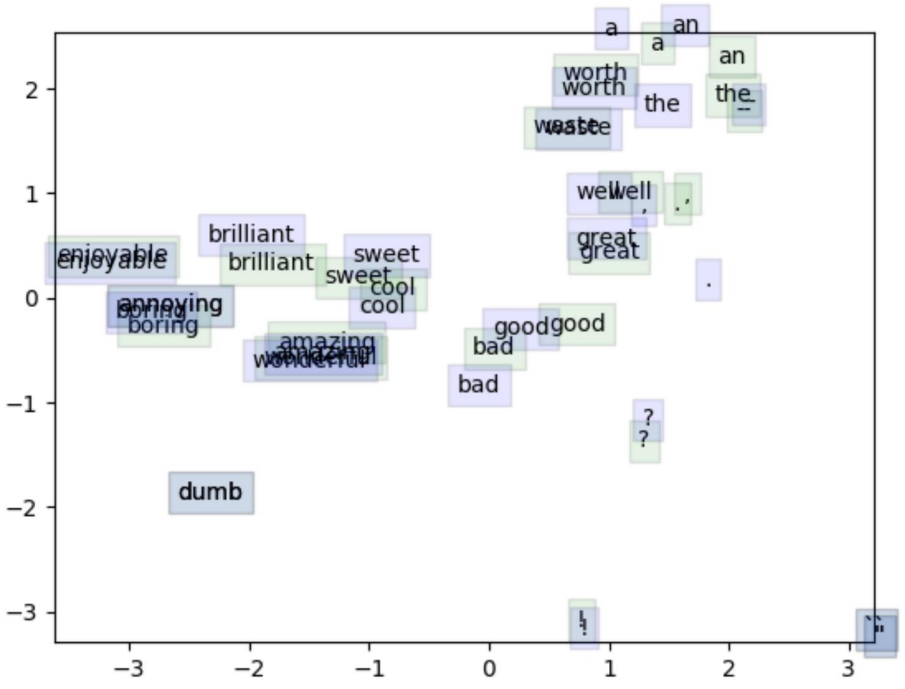

Figure 6: sample retrained word embeddings vs GloVe