

# Probability-Mixing: Semi-Supervised Learning in Question-Answering with Data Augmentation

Stanford CS224N {Default} Project (IID SQuAD Track) default mentor/no sharing

**Dian Huang**

Department of Electrical Engineering  
Stanford University  
dhuang05@stanford.edu

## Abstract

Although back-translation has been effective in text data augmentation, the neural translation system needs to be trained with expensive parallel corpus and can generate noisy samples. Therefore, this paper proposes four simple yet effective text data augmentation without contributing non-negligible label noise. Meanwhile, this paper also addresses false information generation of question answering system and proposes Probability-Mixing, a semi-supervised learning approach that mixes augmented text data. This method improves F1(TP) from 83.58 to 87.51 and reduces the FPR of answerable questions from 0.3 to 0.18. It also improves the EM score from 64.40 to 67.08. Ensembling all method achieves an EM score of 69.96 and an F1 score of 72.40. The best model achieves 64.95/67.79 for EM/F1 score on the test set.

## 1 Introduction

Parts of the rapid development in natural language processing can be attributed to the availability of large datasets. However, the collection of labeled data could be difficult in the field of machine reading comprehension. For example, we need to locate the exact answer position for each question to build a question-answering dataset similar to SQuAD 2.0 [1]. Moreover, a question-answering system could leak private information in the training datasets. Therefore, data augmentation, which generates additional data samples unseen in the training dataset, can largely alleviate these problems. Several prior approaches have adopted data augmentation training a question-answering system. For example, QANet [2] uses back-translation to generate these augmented data samples, but it requires a large amount of parallel corpus to train a machine translation system. [3] uses plausible answers to generate unanswerable questions or as part of the training data, but collecting these labels could be even more expensive.

Furthermore, any of the above data augmentation techniques for question-answering not only slow down the training process with its increase in data size but also lead to more label errors. For example, as demonstrated in QANet, an excessive amount of back-translation data could reduce its performance as it leads to semantic meaning change. Moreover, a good question-answering system should not provide users with false information. For some applications, it is more preferable to give no answer rather than give a wrong answer. However, none of the above work has addressed this issue.

Therefore, we propose four simple data augmentation techniques without contributing to non-negligible label errors. We verify its effectiveness with a total of six different model configurations in this project. Meanwhile, this paper introduces Probability-Mixing, a semi-supervised learning algorithm that generates and mixes text data by a certain probability. This method reduces the FPR from 0.3 to 0.18 and increases F1(TP) by 4 points for the QANet model, which is important to prevent false information generation.

## 2 Related Work

Prior to BERT [4], two models that outperform others are BiDAF[5] and QANet. They both share a similar architecture. Both encode the question and answers then use context-to-query (C2Q) and query-to-context(Q2C) attention mechanism to relate the question with the context and vice versa. Therefore, this work implements these architectures to verify the effectiveness of the proposed methods.

It has been popular to apply semi-supervised learning in the field of computer vision. Recently, there are a few papers that apply this technique to text data. UDA [6] applies semi-supervised learning to reduce the amount of labeled data needed on the text classification task. However, it uses back-translation to preserve the semantics of the original sentences. TextMix [7] mixes the layer parameter of BERT to generate text samples, but it is difficult to control the meaning of text generated.

For semi-supervised learning in computer vision, MixMatch has demonstrated its success in image classification. It reduces the error rate to 11% accuracy on the CIFAR-10 dataset [8] using only 250 labels. MixMatch[9] mixes the labeled images with unlabeled images by summing their pixel values to generate the mixed images. It guesses the labels of unlabeled data using the model from previous epoch and linearly interpolates the labels of the mixed image with the guessed labels and the ground truths. However, for the SQuAD 2.0 dataset, there is no unlabeled data, and it is difficult to mix the text value. Therefore, this work proposes a probability-mixing technique to resolve this problem.

## 3 Approach

### 3.1 Architecture

Both BiDAF and QANet consist of the embedding layer, an encoder layer, a context-query attention (CQT) layer, a modeling layer, and an output layer, as show in figure 5 and figure 6. The difference is that BiDAF uses BiLSTM [10] for its encoder and modeling layer, while QANet uses the convolutional neural network(CNN) and self-attention. Therefore, the QANet uses the same character embedding and the CQT layer from BiDAF. All other blocks are implemented from scratch.

**Embedding Layer:** The embedding layers concatenates the word embedding from GLoVe 300d [11] and the character embedding. For character embedding, with each word truncated or padded to 16 characters, each sequence with length  $L$  is represented as a vector  $X_{seq} \in R^{L \times 16}$ . This vector is then used to look up the tensors  $X_{char_{emb}} \in R^{L \times 16 \times e_{ch}}$  from the character embedding layer with a dimension of  $e_{ch}$ . Next, to group these characters together, a CNN with a kernel size of 5 and output channel  $f = 100$  is applied and yields a tensor  $X_{conv} \in R^{L \times 16 - k + 1 \times f}$ . Finally, the max-pooling layer takes the maximum of the second dimension, which gives vectors  $X_{char_{conv}} \in R^{L \times f}$ . For BiDAF, a variation of the embedding layer is also implemented, which first feeds the input sequence into an absolute position encoder followed by a multi-head self-attention layer [12], then concatenates it with word embedding and character embedding. In this case,  $X_{emb} = \text{concat}([X_{char_{conv}}, X_{word_{emb}}, X_{att}])$ , where  $X_{att}$  is the self-attention output of the input sequence  $X_{seq}$ . Finally, this embedding is fed into a highway network [13].

**Encoder Layer:** The BiDAF uses BiLSTM for the encoder layer. For the QANet, the input goes through a position encoder, a stack of depthwise convolution layers, a multi-head self-attention layer, and layernorm with feedforward. We try both the absolute and the relative position encoder [14]. For absolute position encoder, a sequence of  $L$  word vectors with  $d$  dimension  $X \in R^{L \times d}$  passes through a sinusoidal functions  $PE_{pos,2i} = \sin(pos/10000)^{2i/d}$  and  $PE_{pos,2i+1} = \cos(pos/10000)^{2i/d}$ , where  $pos \in [1, L]$  and  $i$  represents the  $i$ th position of the  $d$  dimensional word embedding. The output is  $X_{pos} = X + PE(pos, i)$ .

Different from the absolute position encoder, the relative position encoder is embedded within the self-attention layer. Let  $x_i$  and  $x_j$  be the  $i_{th}$  and  $j_{th}$  vector of a sequence with length  $L$ . For each attention head, its attention score is calculated as:

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K + a_{ij})}{\sqrt{d_z}} \quad (1)$$

The output of each head  $z_i$  is then calculated as  $z_i = \sum_{j=1}^L \alpha_{ij}(x_j W^V)$ , where  $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^n \exp(e_{ik})}$ .  $W^Q, W^K, W^V$  are the linear transformation matrix that gives more express power to the input vector sequences and  $\alpha_{ij}$  is the  $(i - j)_{th}$  column of the position embedding matrix. Finally, we concatenate all the heads and linearly transform its dimension back to the hidden layer size, which are then fed into a layernorm with feedforward.

**Context-Query Attention Layer (CQT):** Both BiDAF and QANet uses the same context-query attention layer. Its output  $g_i$  is calculated as  $g_i = [c_i; a_i; c_i \odot a_i, c_i \odot b_i] \in R^{8d}$ , where  $c_i, a_i, c_i$  are the  $i_{th}$  context input, context-to-query, and query-to-context attention respectively, and  $d$  denotes the hidden layer size. For the QANet, a projection layer maps this output back to dimension  $d$ .

**Modeling Layer:** The output of context-query attention is then fed into the modeling layer. BiDAF uses BiLSTM for this layer. For QANet, this layer is called Stack Model Encoder Layer in which each output of the layer is repeatedly fed into a stack of 7 Encoded Layers for three times to generate three matrices  $M0, M1$ , and  $M2$ .

**Output Layer:** BiDAF linearly transforms the output of CQT and modeling layer to get the starting position index. For the ending position index, it feeds the modeling layer through a BiLSTM layer similar to its encoder, then linearly transform both the output of CQT and the BiLSTM layer output. For the QANet, the starting position  $P_{start}$  and ending position  $P_{end}$  are calculated using the  $M0, M1, M2$  from the modeling layer as  $P_{start} = \text{softmax}(W1[M0, M1])$ ,  $P_{end} = \text{softmax}(W2[M0, M2])$ .

### 3.2 Data Augmentation

This paper proposes four data augmentation methods: 1. Randomly sample two sentences in a paragraph and connect them with “, and”. This can add one index to the position of the answer and helps the model to match the answer rather than memorizing the answer position. 2. Randomly swap sentence order so that the model can learn to connect sentences from different places together. 3. Delete or replace the sentence that has the answers to make the question unanswerable. (randomly choose one option.) 4. Replace a word for both question and context share. For example, if both question and context have the phrase “United States of America”, we can replace all its occurrence with “United Estates of America”. The substituted word is randomly chosen from words that have a small cosine distance in the GLoVe [11] and do not share the same prefix. This method targets the entity mismatch problem in SQuAD 2.0. Each of the above four methods has a 20% of chance to modify a real sample, leaving only 20% of data to be clean samples in each epoch.

Moreover, the prior works trains with both the modified and the real samples, so it increases the data size and slows down the training process. Therefore, this work discards the old and generate new randomly modified samples for each epoch. As the CPU is used to generate new modified samples while the GPU is used to train the model, this method has minimal impact on the training speed.

The baseline model is the BiDAF without character embedding. Applying these methods increase its EM and F1 score of from 57.9 to 60.94 and from 61.44 to 63.88 respectively, as shown in table 2.

### 3.3 Probability-Mixing

With data augmentation, the model can no longer memorize the position index, so achieves a higher score in all categories. However, it is still possible that the model memorizes the words around an answer rather than understanding the context. To prevent the model from memorizing the context around, we can replace some of the words with similar words found in GLoVe. However, replacing a word sometimes changes the meaning of a sentence, as the similar word found at word vector space usually share similar context rather than similar meaning. Therefore, this work proposes a semi-supervised learning method to address this problem by assuming these augmented texts as unlabeled data.

**Label Guessing:** We need to prepare three types of data for this semi-supervised problem: labeled data sample  $X^{(i)}$ , unlabeled data sample  $U^{(i)}$ , and the mixed data sample  $M^{(i)}$ . We denote  $i$  to be the  $i_{th}$  sample of a batch and  $j_{th}$  to be the  $j_{th}$  word of a sequence,  $y$  to be the ground-truth position,  $z$  to be the interpolated label, and  $q$  to be the pseudo-label guessed by the model for the following discussion of probability-mixing. To generate unlabeled data sample  $U^{(i)}$  for the SQuAD 2.0, each

word  $w_j^{label}$  in both the question and context of the labeled data sample  $X^{(i)}$  has a probability  $p \sim \text{uniform}(0, 1)$  to be replaced with a similar word  $w_j^{unlabel}$  found by GLoVe. Then we use the QANet model  $f_{model}$  with sharpening to guess their starting position  $q_{start}^{(i)}$  and ending position  $q_{end}^{(i)}$  as follows:

$$p_{start_{guess}}^{(i)}, p_{end_{guess}}^{(i)} = f_{model}(U^{(i)}) \quad (2)$$

$$q_{start}^{(i)} = \text{sharpen}(p_{start_{guess}}^{(i)}, T), q_{end}^{(i)} = \text{sharpen}(p_{end_{guess}}^{(i)}, T) \quad (3)$$

The sharpening function can be represented as  $\text{sharpen}(p, T)_j := p_j^{1/T} / \sum_{k=1}^L p_k^{1/T}$ . As temperature  $T$  approaches 0, the sharpening function will produce a one-hot function. It steers the classifier's decision boundary from avoid passing through high-density regions of the samples when separating different classes.

**Probability-Mixing:** MixMatch generates mixed images by summing the pixel value between labeled and unlabeled images then linearly interpolates their labels, which does not apply to our text data. Therefore, this work replaces the summation of pixel value with probability in which each word  $w_j^{mix}$  in  $M^{(i)}$  has a probability  $\lambda \sim \text{Beta}(\alpha, \alpha)$  to be word  $w_j^{label}$  from  $X^{(i)}$  and  $1 - \lambda$  to be word  $w_j^{unlabel}$  from  $U^{(i)}$ , where  $\alpha$  is a hyperparameter set to be 0.75.

**Label Interpolation:** The labels of mixed data are linearly interpolated as follows:

$$z_{start}^{(i)} = \lambda * y_{start}^{(i)} + (1 - \lambda) * q_{start}^{(i)} \quad (4)$$

$$z_{end}^{(i)} = \lambda * y_{end}^{(i)} + (1 - \lambda) * q_{end}^{(i)} \quad (5)$$

where  $y_{start}^{(i)}, y_{end}^{(i)}$  are the ground truth starting and ending index of labeled data  $X^{(i)}$

**Loss:** Then both the labeled and the mixed data are fed into the model  $f_{model}$  as:

$$p_{start}^{(i)}, p_{end}^{(i)} = f_{model}(X^{(i)}) \quad (6)$$

$$p_{start_{mix}}^{(i)}, p_{end_{mix}}^{(i)} = f_{model}(M^{(i)}) \quad (7)$$

The mixed outputs are split into two part, with one for the the cross entropy loss calculation and the other for the mean square loss calculation.

$$L_{cross} = \frac{1}{|B|} \left( \sum_{i=0}^B y_{start}^{(i)} \log p_{start}^{(i)} + \sum_{i=0}^B y_{end}^{(i)} \log p_{end}^{(i)} + \sum_{i=0}^{B/2-1} z_{start}^{(i)} \log p_{start_{mix}}^{(i)} + \sum_{i=0}^{B/2-1} z_{end}^{(i)} \log p_{end_{mix}}^{(i)} \right) \quad (8)$$

$$L_{sq} = \frac{1}{|B/2||L|} \left( \sum_{i=B/2}^B W(p_{start_{mix}}^{(i)} - q_{start}^{(i)})^2 + W(p_{end_{mix}}^{(i)} - q_{end}^{(i)})^2 \right) \quad (9)$$

$$L_{total} = L_{cross} + \lambda_w \lambda_U L_{sq} \quad (10)$$

where  $B$  is the batch size,  $L$  is the maximum sequence length for a batch of data,  $\lambda_w$  is linear warm up that reaches 1 at epoch 12, and  $\lambda_U = 600$ .  $W$  is the weight set to compensate the imbalance of position index, as a large portion of the labels are 0, which means the question is not answerable. As shown in table 1, The weights for  $QANet_{semi}$  are all set to be 1, and the weights for  $QANet_{semi-w}$  are set to 1 except label 0, which is set to be 0.5. Note that for this work, the mixed data batch is split into two half for the cross-entropy and mean-square loss. More experiments would be needed to find an more optimal split-up.

## 4 Experiments

For ease of discussion, we denote the name for each model configuration as in table 1. For example,  $QANet_{small}^{absolute}$  means it is a small-size QANet model using absolute position encoding.

| Aritecture                                      | Number of Parameters | Run time /epochs | Hidden size | Feedforward | Self-Attention Heads | Description                                                                                   |
|-------------------------------------------------|----------------------|------------------|-------------|-------------|----------------------|-----------------------------------------------------------------------------------------------|
| <i>BiDAF</i>                                    | 1.4M                 | 7 mins           | 100         |             |                      | Baseline                                                                                      |
| <i>BiDAF<sup>char</sup></i>                     | 1.7M                 | 10 mins          | 100         |             |                      | <i>BiDAF</i> + character embedding                                                            |
| <i>BiDAF<sup>att</sup></i>                      | 2M                   | 12 mins          | 100         | 48          | 4                    | <i>BiDAF<sup>char</sup></i> + self-attention embedding                                        |
| <i>QANet<sup>absolute</sup><sub>small</sub></i> | 1M                   | 30 mins          | 96          | 48          | 4                    | <i>QANet</i> with absolute position encoding                                                  |
| <i>QANet<sup>relative</sup><sub>small</sub></i> | 1.4M                 | 35 mins          | 96          | 48          | 4                    | <i>QANet</i> with relative position encoding                                                  |
| <i>QANet<sup>relative</sup><sub>large</sub></i> | 2.1M                 | 45 mins          | 128         | 32          | 8                    | <i>QANet<sup>relative</sup><sub>small</sub></i> with more heads and hidden layers             |
| <i>QANet<sub>semi</sub></i>                     | 1.4M                 | 70 mins          | 96          | 48          | 4                    | <i>QANet<sup>relative</sup><sub>small</sub></i> with p-mix                                    |
| <i>QANet<sub>semi-w</sub></i>                   | 1.4M                 | 70 mins          | 96          | 48          | 4                    | <i>QANet<sub>semi</sub></i> with 0.5 weight on non-answerable class for the square loss term. |

Table 1: Configurations for different models. Note that *QANet<sub>semi</sub>*, *QANet<sub>semi-w</sub>*, and *QANet<sup>relative</sup><sub>small</sub>* are using the same model but with different loss or training methods.

#### 4.1 Data and Experimental Details

The SQuAD 2.0 dataset has a total of 130,319 training samples and 6078 dev samples. Each passage is truncated to a maximum length of 400 words. With a Tesla T4 GPU, all BiDAF models were trained for 30 epochs, and all QANet models were trained for 40 epochs. There is no noticeable difference in run time when using data augmentation, as the CPU is doing the text processing. The BiDAF-based models use a batch size of 64 and the Adadelata optimizer with a learning rate of 0.5. The QANet-based models use a batch size of 24 and the Adam optimizer with a learning rate of 0.1. The dropout for word embedding, character embedding, and others are 0.2, 0.05, and 0.1 respectively for all the models. *QANet<sub>semi</sub>* and *QANet<sub>semi-w</sub>* take 70 minutes per epoch as each time the model needs to process labeled, mixed, and unlabeled data. Moreover, BiDAF is much faster than QANet even though they have a similar number of parameters. One possible reason is that QANet’s CNN requires more GPU memory to be parallelized.

#### 4.2 Evaluation method

It is important to avoid the model from outputting false information. For some applications, it is more preferable that the model does not know the answer rather than gives user the wrong answer. Therefore, besides EM/F1, two metrics, EM(TP) and F1(TP) are used to measure how well the model locates the answer when it predicts if a question is answerable correctly, which are the exact match and F1 score for only the true positive cases of the answerable questions. We also use the False Positive Rate (FPR), which is calculated as  $FPR = \frac{FP}{FP+TN}$  to measure how likely the model predicts an unanswerable question as answerable, where *FP* and *TN* are the false positive and true negative number of predicting a question as answerable (positive = answerable).

#### 4.3 Results

As shown in Table 2, using data augmentation improve scores in all category, as it provides extra information for the model to learn. The F1 scores of the QANet models are at least three points higher than those of BiDAF. This may indicate that the self-attention structure is more effective in context matching. The reason might be that it focuses less on the sequential order of words and more on the global relationship among words. Interestingly, using relative position encoding also focuses more on the relative distance between words rather than their sequential order, so it scores about 1 point better in F1 than using absolute position encoding, which still retains the sequential information of the input with the absolute position of a sequence.

It is worth noting that data augmentation does not improve the F1 score for the BiDAF architecture with self-attention embedding as much as other BiDAF configurations. It improves the F1 score from 64.09 to 65.47, but with only character embedding, the F1 score improves from 63.17 to 65.76. A similar phenomenon also occurs with all the *QANet<sub>small</sub>* models, where the data augmentation

| Methods                                         | No Data Augmentation |              |              |              |              |             | With Data Augmentation |              |              |              |              |             |
|-------------------------------------------------|----------------------|--------------|--------------|--------------|--------------|-------------|------------------------|--------------|--------------|--------------|--------------|-------------|
|                                                 | EM                   | F1           | AvNA         | EM(TP)       | F1(TP)       | FPR         | EM                     | F1           | AvNA         | EM(TP)       | F1(TP)       | FPR         |
| <i>BiDAF</i>                                    | 57.90                | 61.44        | 67.64        | 74.45        | 82.91        | 0.44        | 60.94                  | 63.88        | 69.38        | 76.01        | 84.33        | 0.35        |
| <i>BiDAF<sup>char</sup></i>                     | 59.82                | 63.17        | 69.48        | 75.03        | 83.63        | 0.41        | 62.52                  | 65.76        | 71.27        | 75.27        | 84.37        | 0.32        |
| <i>BiDAF<sup>att</sup></i>                      | 60.64                | 64.09        | 70.60        | 74.20        | 83.08        | 0.39        | 62.49                  | 65.47        | 71.19        | 75.89        | 84.10        | 0.33        |
| <i>QANet<sup>absolute</sup><sub>small</sub></i> | 63.17                | 66.86        | 73.28        | 12.33        | 13.55        | 0.33        | 64.87                  | 67.80        | 73.02        | 74.94        | 83.93        | 0.23        |
| <i>QANet<sup>relative</sup><sub>small</sub></i> | 64.40                | 68.25        | 74.40        | 74.41        | 83.58        | 0.30        | 65.81                  | 69.33        | 74.40        | 74.59        | 84.27        | 0.26        |
| <i>QANet<sup>relative</sup><sub>large</sub></i> | 64.69                | 68.18        | 74.22        | 74.71        | 83.94        | 0.30        | <b>67.52</b>           | <b>70.80</b> | <b>76.13</b> | 76.18        | 85.22        | 0.24        |
| <i>QANet<sub>semi</sub></i>                     |                      |              |              |              |              |             | 66.45                  | 68.75        | 72.38        | 79.69        | 87.51        | 0.18        |
| <i>QANet<sub>semi-w</sub></i>                   |                      |              |              |              |              |             | <b>67.08</b>           | <b>69.87</b> | <b>73.92</b> | <b>78.56</b> | <b>87.29</b> | <b>0.20</b> |
| <b>Ensemble</b>                                 | <b>69.91</b>         | <b>72.40</b> | <b>76.39</b> | <b>81.54</b> | <b>88.59</b> | <b>0.21</b> |                        |              |              |              |              |             |

Table 2: Results for all model configurations on the dev set. Note that models with data augmentation outperform in all categories. QA-semi has a higher EM score for the same model configuration. Relative position encoding has slightly better performance than absolute ones. *QANet<sub>semi</sub>* has substantially lower FPR than all other architecture. The best model *QANet<sup>relative</sup><sub>large</sub>* achieves **64.95/67.79** on EM/F1 for the test set.

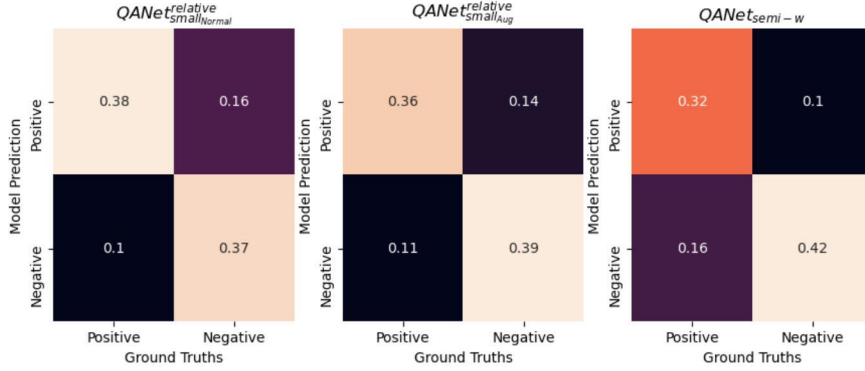


Figure 1: Confusion matrix for *QANet<sub>small</sub>* with different training methods. *QANet<sub>semi-w</sub>* gives the lowest false positive but worst false negative, as it tends to turn away those harder questions.

improves the F1 score by less than 1 point. All these models are using self-attention layers with fewer heads. As a result, these models may not have sufficient expressive power to learn the augmented data. On the contrary, data augmentation has more impact on the *QANet<sub>large</sub>*, which uses an 8-head self-attention layer. The BiDAF architecture also benefits more from data augmentation. Therefore, to have similar expressive power on text data, we need much more parameters for the self-attention layer than recurrence neural network (RNN), as the latter incorporates more sequential information of the input.

Similar reasoning also explains why QANet does not suffer from overfitting as much as BiDAF does, as shown in figure 7. Different from RNN, the CNN and self-attention can only rely on the positional encoder to provide some amount of positional information, so they have less capacity in memorizing the position index of an answer. Similarly, figure 7 also shows that absolute position encoding suffers more from overfitting than relative position encoding, as it still retains the sequential order information using the absolute position indices of tokens. These experiments demonstrate that sequential order information is not beneficial to query-context matching. Using data augmentation to shift the indices of the tokens alleviates this problem. The result in table 2 also shows that QANet has a higher AvNA score and lower FPR than BiDAF. So QANet performs much better in determining if a question is answerable, the reason of which might be that QANet tries to match the context rather than the position to check if similar contexts are present for both question and answer. However, QANet shows no advantage on EM(TP) and F1(TP) score, as demonstrated in table 2. A possible reason is that both models can still try to match the context to the answer words in their memory when looking for the location of the answer.

|                                     | What      | Who       | Where     | How       | When      | Which     | Why       |
|-------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Total                               | 3726      | 714       | 275       | 605       | 561       | 274       | 92        |
| $QANet_{semi-w}$                    | 67.7/70.4 | 68.3/70.7 | 66.2/69.9 | 64.3/67.3 | 71.5/73.3 | 58.4/61/2 | 52.2/60.9 |
| $QANet_{small}^{relative_{normal}}$ | 64.1/67.9 | 66.9/69.1 | 60.0/64.4 | 61.2/66.9 | 69.7/71.9 | 66.8/71.3 | 59.8/67/4 |
| $QANet_{small}^{relative_{aug}}$    | 65.8/69.3 | 69.3/71.8 | 64.0/67.8 | 62.9/67.4 | 69.5/72.2 | 61.7/64.3 | 58.7/68.1 |

Table 3: The table shows the EM/F1 score for different type of questions.  $QANet_{small}^{relative_{normal}}$  is  $QANet_{small}^{relative}$  that does not use data augmentation.  $QANet_{small}^{relative_{aug}}$  uses data augmentation.

$QANet_{semi}$  adopts a semi-supervised approach and resolves this problem by replacing words with others. As a result, it has a higher EM score and much higher EM(TP) and F1(TP) scores for the same configuration. Data augmentation improves the EM(TP) and F1(TP) of  $QANet_{small}^{relative}$  from 74.41 and 83.58 to 74.59 and 84.27.  $QANet_{semi}$  further improves them to 79.69 and 87.51. More importantly, with both data augmentation and probability-mixing,  $QANet_{semi}$  reduces FPR from 0.3 to 0.18, making it a much better model in avoiding false information generation.

However,  $QANet_{semi}$  has a lower AvNA score. The reason is that the mean-square loss term can be zero when the model predicts all questions to be non-answerable. Therefore, giving less weight to the non-answerable label in the mean-square loss alleviates this problem, as demonstrated by the performance of model  $QANet_{semi-w}$ , where the mean-square loss for the non-answerable index is weighted by 0.5. This method improves the EM, F1, and AvNA score from 66.45, 58.75, and 72.38 to 67.08, 69.87, and 73.92 respectively.

**Ensembling:** Finally, all the models are ensembled together with the majority voting method. We first pick the answer with a majority of votes. If there is a tie, we check if it ties between the non-answerable question and the answer position. In this case, we will pick the answer from the model with the highest overall F1-score. However, if it is a tie among answer positions, we will choose from the model with the highest F1(TP) score. The score for EM and F1 are 69.5 and 72.3 on the dev set and 66.5 and 69.2 on the test set.

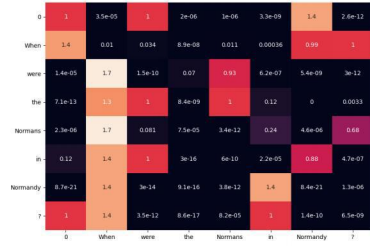
## 5 Analysis

As shown in table 3,  $QANet_{semi-w}$  significantly improves the F1 score for context matching question that has word such as "what", "who", "where", "when", and "how" (many). However, it does poorly at questions with logical reasoning, which typically starts with "why". Two reasons account for this. First, none of the data augmentation methods discussed in this paper provides additional information on logical reasoning for the model to learn. Second, as shown in equation 10, for hard questions where context matching is difficult to find, guessing them as non-answerable typically has a better chance of being correct than guessing among all the position indices. So this model simply turns away from those hard questions. Therefore,  $QANet_{semi-w}$  has the lowest false positive but highest false negative ratio compared to other  $QANet_{small}$  of the same model configuration, as shown in figure 1

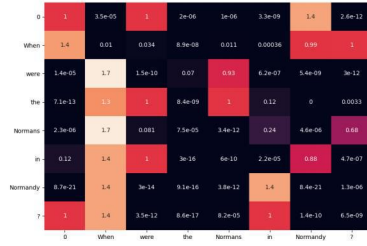
Moreover, using data augmentation does not perform well on the questions with the word "which". One possible reason is that merging a sentence, deleting sentences, and replacing a word are not related to choosing among several options. The other reason is that the training loss is slightly higher for using data augmentation, as shown in figure 7, so most of the gradient update targets at the matching of context, leaving less room to optimize for the "which" question.

We also compared the self-attention weight among the  $QANet_{small}$  architectures. As shown in figure 2, the self-attention weight of  $QANet_{semi}$  concentrates on the word "when" with others.  $QANet_{semi-w}$  concentrates mostly on "when" and "were" with "Normans" and "Normandy". For  $QANet_{small}^{relative_{aug}}$ , it is more scattered, focusing on the attention of "When", "were", "Normans", "Normandy" with almost all other words. The attention weight of "QA-small" is most scattered. Therefore,  $QANet_{semi}$  and  $QANet_{semi-w}$  can match the key information between the question and the context better than others, making it finding the location of the answer more accurately.

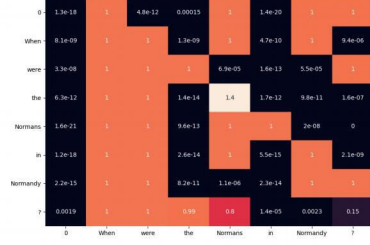
Two examples highlight the strength and weaknesses of the  $QANet_{semi-w}$  model. For the first example, the self-attention weight shows that both  $QANet_{semi}$  and  $QANet_{small}^{relative_{normal}}$  correctly



(a) Self-attention weight for  $QANet_{semi}$



(b) Self-attention weight for  $QANet_{semi-w}$



(c) Self-attention weight for  $QANet_{small}^{relative_{aug}}$



(d) Self-attention weight for  $QANet_{small}^{relative_{normal}}$

Figure 2: The self-attention weight for  $QANet_{semi}$  concentrates on fewer words.

identifies the word "country" as the keyword. However, they both predict "Germany" as the answer since the phrase "empire allied with Germany" is more similar to the question, which shows that the model focuses more on query-context matching and lacks inference ability. The second example shows how data augmentation and probability-mixing improve the model's ability to identify the keywords. The keywords of the question are "nickname" rather than "known" and "public".  $QANet_{semi-w}$  concentrates its weight on "name" and "Isiah Bowman" and less on the word "public", so it correctly predicts the answer. QA-normal scatters its weight around "Isiah", "name", "known", and "public", so falsely predicts "American Empire" as the answer.

## 6 Conclusion

Our experiments demonstrate that sequential order information does not necessarily help query-context matching, and excessive sequential order information in BiDAF's RNN can lead to overfitting. To alleviate overfitting and add more variety to the training samples, we propose four data augmentation methods without introducing non-negligible label noise, which improves the F1 scores of BiDAF and the QANet with 8 heads by at least 2 points. We also propose the Probability-Mixing method to prevent the model from memorizing the context, which significantly improves its ability in query-context matching. This method reduces the FPR from 0.3 to 0.18 and increases F1(TP) by 4 points for the QANet model, making it a much better model in preventing the generation of misleading information for the question-answering system.

Due to limited hardware resources, it is difficult to test our Probability-mixing approach on QA-large, as each epoch takes around two hours on a Tesla T4 GPU. In the future, we hope to test Probability-mixing on larger models. Moreover, both the proposed data augmentation and the probability-mixing method are only limited to the query-context matching problem. We hope to address problems involved with logical reasoning in the future.

## References

- [1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018.

- [2] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.
- [3] Haichao Zhu, Li Dong, Furu Wei, Wenhui Wang, Bing Qin, and Ting Liu. Learning to ask unanswerable questions for machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4238–4248, Florence, Italy, July 2019. Association for Computational Linguistics.
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
- [5] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.
- [6] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training, 2020.
- [7] Jiaao Chen, Zichao Yang, and Diyi Yang. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification, 2020.
- [8] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).
- [9] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *CoRR*, abs/1905.02249, 2019.
- [10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.
- [12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.
- [13] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.
- [14] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations, 2018.

## A Appendix

### A.1 Examples

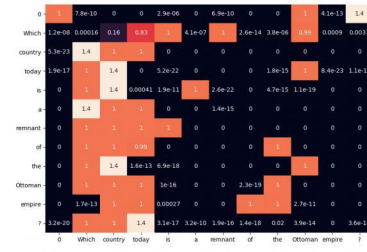
**Question:** Which country today is a remnant of the Ottoman empire?

**Context:** With Istanbul as its capital and control of lands around the Mediterranean basin, the Ottoman Empire was at the center of interactions between the Eastern and Western worlds for six centuries. Following a long period of military setbacks against European powers, the Ottoman Empire gradually declined into the late nineteenth century. The *empire allied with Germany* in the early 20th century, with the imperial ambition of recovering its lost territories, but it dissolved in the aftermath of World War I, leading to the emergence of the new state of **Turkey in the Ottoman Anatolian heartland**, as well as the creation of modern Balkan and Middle Eastern states, thus ending Turkish colonial ambitions.

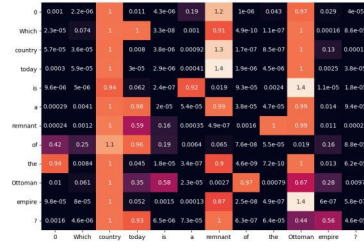
**Prediction:**

$QANet_{semi-w}$ : Germany

$QANet_{semi-w}^{relative_{normal}}$ : Germany



(a) Self-attention weight for  $QANet_{semi-w}$



(b) Self-attention weight for  $QANet_{small}^{relative_{normal}}$



(a) Self-attention weight for  $QANet_{semi-w}$



(b) Self-attention weight for  $QANet_{small}^{relative_{normal}}$

Figure 4: The self-attention of  $QANet_{semi-w}$  concentrates on fewer keywords.

**Ground Truth:** Turkey

**Question:** What was Isiah Bowman nick name, as known by the public.

**Context:** One key figure in the plans for what would come *to be known as American Empire*, was a geographer named Isiah Bowman. Bowman was the director of the American Geographical Society in 1914. Three years later in 1917, he was appointed to then President Woodrow Wilson’s inquiry in 1917. The inquiry was the idea of President Wilson and the American delegation from the Paris Peace Conference. The point of this inquiry was to build a premise that would allow for U.S authorship of a ‘new world’ which was to be characterized by geographical order. As a result of his role in the inquiry, **Isiah Bowman would come to be known as Wilson’s geographer.**

$QANet_{semi-w}$ : Wilson’s geographer

$QANet_{semi-w}^{relative_{normal}}$ : American Empire

**Ground truth:** Wilson’s geographer

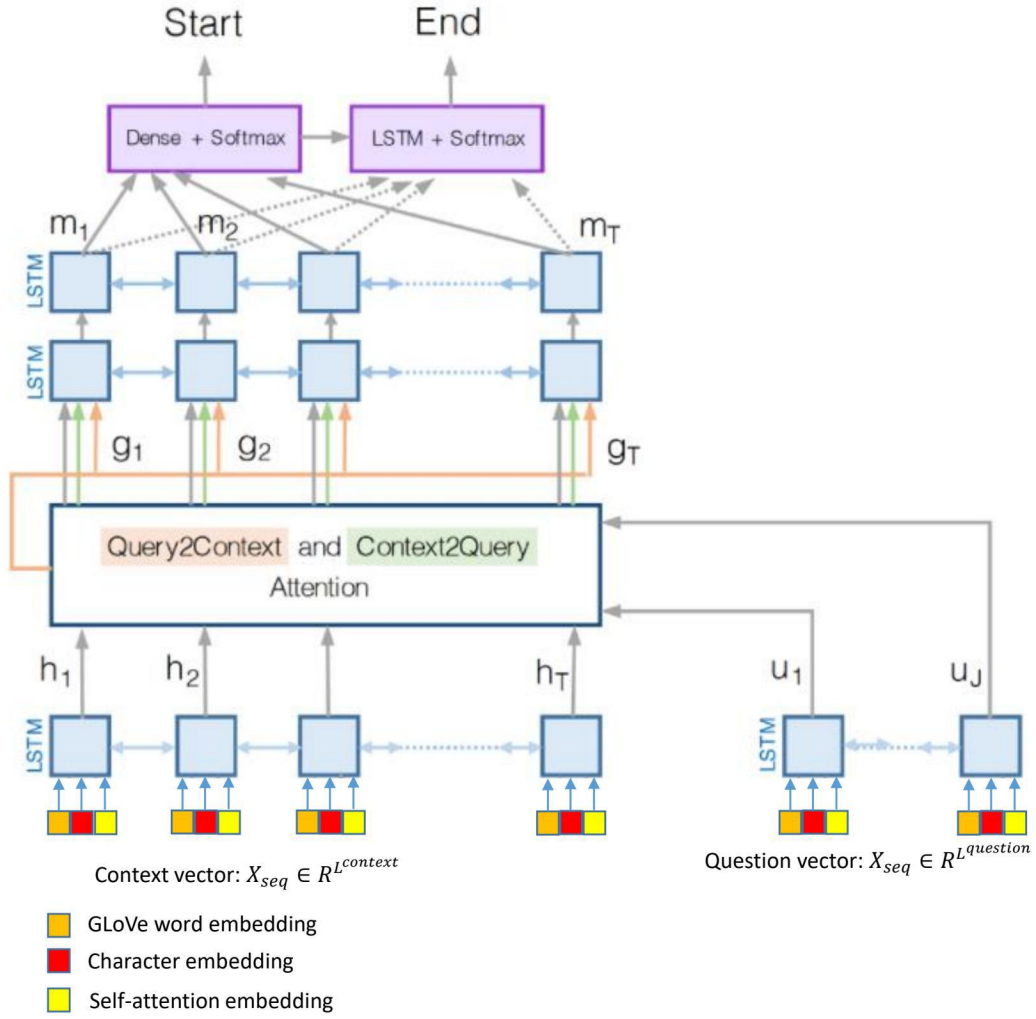


Figure 5: Figure modified from [5]. Additional self-attention embedding concatenated with GloVe and character embedding.

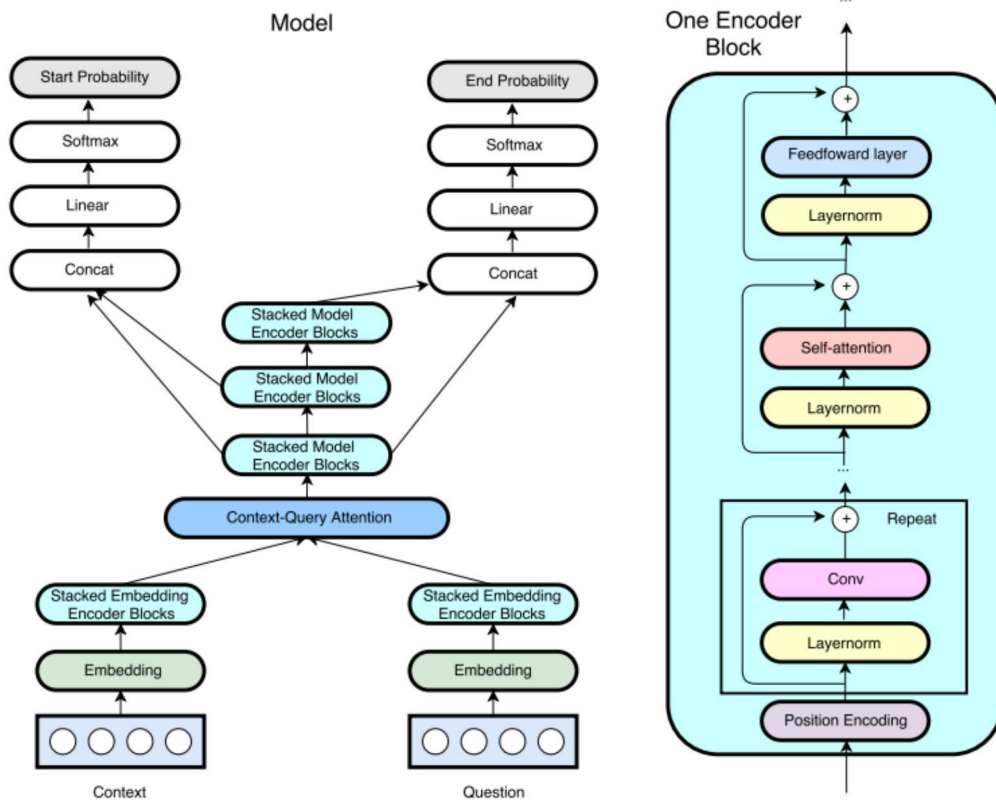
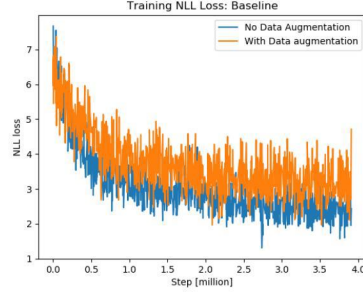
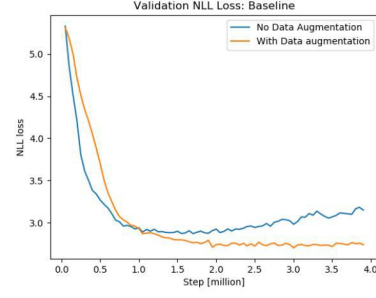


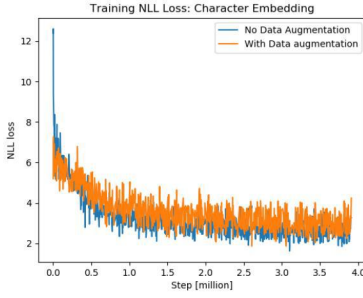
Figure 6: Figure from [2]. Additional self-attention embedding concatenated with GLoVe and character embedding.



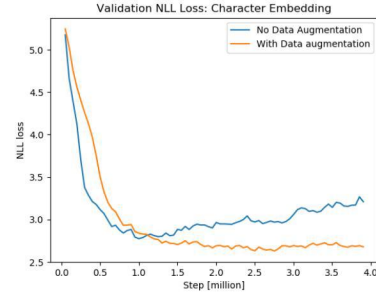
(a) Training:  $BiDAF$



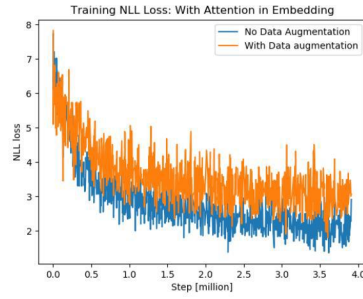
(b) Validation:  $BiDAF$



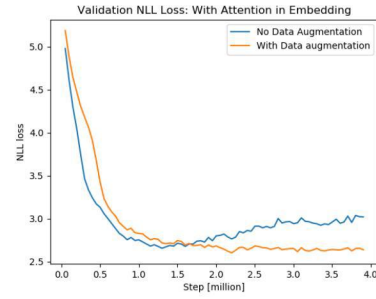
(c) Training:  $BiDAF^{char}$



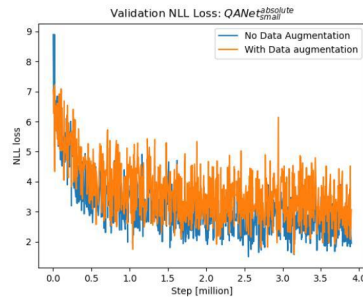
(d) Validation:  $BiDAF^{char}$



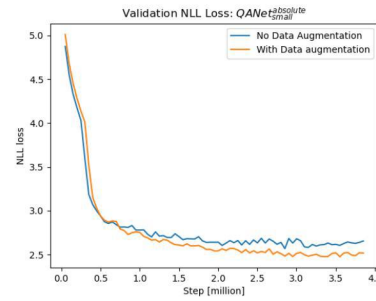
(e) Training:  $BiDAF^{att}$



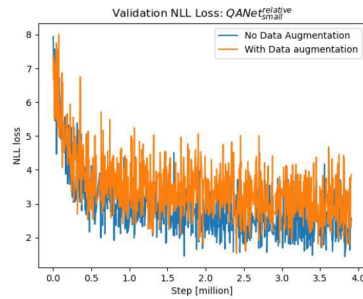
(f) Validation:  $BiDAF^{att}$



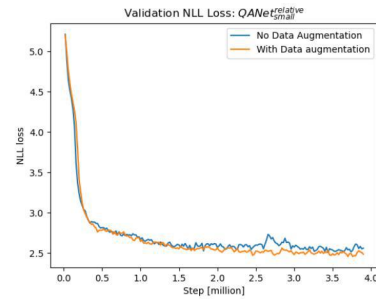
(g) Training:  $QANet_{small}^{absolute}$



(h) Validation:  $QANet_{small}^{absolute}$



(i) Training:  $QANet_{small}^{relative}$



(j) Validation:  $QANet_{small}^{relative}$

Figure 7: For all models, without data augmentation, the validation NLL loss starts increasing at about 1 million steps, but with data augmentation the validation NLL loss remains decreasing.