Improving the Performance of Previous QA Models

Stanford CS224N Default Project

Yen-Yu Chang
Department of Electrical Engineering
Stanford University
yenyu@stanford.edu

Cheng-Min Chiang
Department of Computer Science
Stanford University
cmchiang@stanford.edu

Abstract

Question answering is a challenging problem that tests language processing models the ability to comprehend natural languages. In this project, we implemented two models, BiDAF and QANet, to solve the Stanford question answering dataset (SQuAD) 2.0. We experienced different methods to improve the performance of these models, including adding character embedding layers, data augmentation, and ensemble modeling. Finally, we compared the result across different experiments and gave an analysis of our models. In the end, our best model achieved **F1/EM score of 68.71/65.38** in the test leaderboard.

1 Key Information to include

- Mentor: Chris Waites
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

Question answering is a challenging task that tests a model's ability to comprehend the language. It also has many potential applications, including natural language search engines, AI chatbots, and many more. In this project, we worked on the SQuAD 2.0 dataset. SQuAD is a reading comprehension dataset consists of articles extracted from Wikipedia and 100,000+ question-answer pairs. For each question, there is a corresponding reading paragraph. The goal is to find a span of text in the context paragraph that answers the question, and the model will not need to generate an answer sentence itself. SQuAD 2.0 includes questions that cannot be answered from the paragraph, which prevents a model to simply find a span that is most related to the question. The model will need to truly comprehend the question and the paragraph in order to succeed on this task.

We implemented two models to solve this problem, BiDAF [1] and QANet [2]. We extended the baseline BiDAF model by adding character embedding layers to the model. We also implemented QANet and tested the performance of the model on different parameters. Then, we experimented with different methods to boost their performance. We created an additional dataset with 79674 more questions by using back-translation on the available training data. We tested how our model performed on this augmented dataset. The other method we adopted is ensemble modeling. We selected a total of 5 different models we trained with different parameters and combined their results.

3 Related Work

3.1 Models solving SQuAD

Machine reading comprehension and question answering has become an important topic in the NLP domain. Since the release of the SQuAD question answering dataset, several models have been proposed to tackle this task. These models roughly fall into two categories, models based on recurrent networks or those based on self-attention layers. Recurrent networks (e.g., LSTM) are a common way in NLP dealing with contextual information and the interaction between pair of words. Models that adopt this method include BiDAF [1], R-Net [3], DCN [4]. Inspired by Transformer [5], there are also proposed models that use self-attention with convolution layers instead of recurrent layers. These models are shown to have performance as good as recurrent ones and enjoy the advantage of faster computation as they do not suffer from long dependency chains in recurrent layers.

Recently, pre-trained models have shown success in various NLP tasks, which include reading comprehension. In fact, pre-trained models, such as ALBERT [6], have dominated the leader board of the SQuAD 2.0 dataset. They are able to achieve an F1 score above 90, which the F1 score of non-pretrained models typically falls between the range of 60 – 75. In this project, we are not allowed to use pre-training methods, so we do not consider these models.

3.1.1 Data augmentation

Data augmentation tries to create more training data from a limiting data set. In NLP, one popular method called back translation augments the data by translating the texts into another language and then translate them back. The resulting texts will be slightly different from the original ones, which makes them good candidates for being additional training data.

Easy Data Augmentation [7] is another data augmentation method for NLP tasks. For sentences in the training set, we randomly apply one operation from four different kinds of operations:

1. **Synonym Replacement**: Randomly choose n words and replace each of them with one of its synonyms.
2. **Random Insertion**: Insert a synonym of a random word in the sentence. Repeat the process n times.
3. **Random Swap**: Randomly choose two words and swap their positions. Repeat the process n times.
4. **Random Deletion**: Randomly delete each word with probability p.

These new sentences are added back to the dataset. The authors showed that they were able to boost the performance of models on text classification tasks by using this technique.

4 Approach

4.1 Models

First, we implement two models, BiDAF and QANet, to solve the task. They are similar in structure and are both formed by 6 different layers. In the following, let $\boldsymbol{c} = [c_1, c_2, \ldots, c_n]$ be the words in the context (reading passage), and $\boldsymbol{q} = [q_1, q_2, \ldots, q_m]$ be the words in the question, and h be the hidden size. The duty of each layer is roughly described as follows:

1. **Input Embedding Layer** maps each word into a feature vector. We apply this layer on both the context and the query to get $\boldsymbol{c}_{\text{emb}} \in \mathbb{R}^{n \times h}$ and $\boldsymbol{q}_{\text{emb}} \in \mathbb{R}^{n \times h}$.
2. **Embedding Encoding Layer** scans through the entire text and refines each word embedding by utilizing contextual clues from the surrounding word. Both $\boldsymbol{c}_{\text{emb}}$ and $\boldsymbol{q}_{\text{emb}}$ are fed into this layer to get the matrix $\boldsymbol{c}_{\text{enc}}$ and $\boldsymbol{q}_{\text{enc}}$, which consist of feature vector for each word in the context and the question, respectively.
3. **Context-Query Attention Layer** is crucial in the question-answering task because it allows the model to couple the information from both the context and the question. $\boldsymbol{c}_{\text{emb}}$ and $\boldsymbol{q}_{\text{emb}}$ will be fed to this layer at the same time, which is different from previous layers where we process data from the context and the question separately.

   Both BiDAF and QANet use attention methods to combine the information from $\boldsymbol{c}_{\text{emb}}$ and $\boldsymbol{q}_{\text{emb}}$. This information is used to produce the feature vector for each word in the context.
4. **Model Encoding Layer** further refines the feature vector from the output of the context-query attention layer.

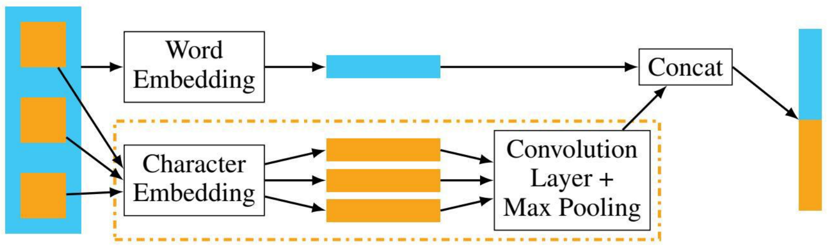

Figure 1: The structure of the embedding layer. The orange block is the character embedding layer we added.

5. **Output Layer** makes predictions of the probability of each position being the start or the end of the answer span based on the feature vectors. Specifically, this layer will output $\boldsymbol{p}_s$ and $\boldsymbol{p}_e$, where $\boldsymbol{p}_{s,i}$ and $\boldsymbol{p}_{e,i}$ are the log-probability of the i-th position being the start and the end of the span, respectively.

4.1.1 BiDAF

Bi-Directional Attention Flow (BiDAF) [1] is a model that achieved state-of-the-art performance on the SQuAD v1.1 dataset [8] in 2016. An implementation of BiDAF is included in the starter code, and we describe the structure of BiDAF here.

1. **Input Embedding Layer**: Figure 1 shows the structure of the layer. The input embedding layer assigns each word in the text with an embedding vector. The word embedding is initialized to pre-trained GloVe [9] vectors with dimension 300, and these vectors are fixed during training. In addition to the word embedding, we implemented a character embedding layer. Each character in a word is mapped to a trainable vector in $\mathbb{R}^D$ where $D = 200$. We pad or truncate characters in a word so that each word has a fixed number of $L = 16$ characters. After the character embedding look-up, we get a tensor of size $\mathbb{R}^{L\times D}$. Then, the tensor is passed to a 1-D convolution on the first dimension with D input and output channels, and we apply a max-pooling to the result. Finally, we concatenate the vectors from word embedding and character embedding to get a vector of 500 dimension. This vector is then passed to a two-layer highway network [10].
2. **Embedding Encoding Layer**: BiDAF uses a bi-directional LSTM in this layer to encode the text.
3. **Context-Query Attention Layer**: This layer is described thoroughly in the project handout so we only give a high-level desciption. Let $\boldsymbol{c}_{\text{emb},i}$ and $\boldsymbol{q}_{\text{emb},j}$ be the feature vector of the i-th word and the j-th word in the context and the query, respectively. First, we calculate the similarity matrix $S_{i,j} = \boldsymbol{w}^\top[\boldsymbol{c}_{\text{emb},i}; \boldsymbol{q}_{\text{emb},j}; \boldsymbol{c}_{\text{emb},i} \circ \boldsymbol{q}_{\text{emb},j}]$, and then we use $S_{i,j}$ to comute context-to-query and query-to-context attention vector $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ for each word. The output of this layer is $[\boldsymbol{c}_{\text{emb},i}; \boldsymbol{a}_i; \boldsymbol{c}_{\text{emb},i} \circ \boldsymbol{a}_i; \boldsymbol{c}_{\text{emb},i} \circ \boldsymbol{b}_i]$.
4. **Model Encoding Layer**: A two-layer bi-direction LSTM is used in this layer.
5. **Output Layer**: The feature vectors will be fed into another bi-directional LSTM and then a linear layer. The final output is the log-probability vectors.

4.1.2 QANet

QANet [2] is another model that achieved state-of-the-art (with data augmentation) in the SQuAD 1.1 dataset at the time of publishing. Inspired by Transformer model, it uses self-attention and convolution layers to eliminate the needs of recurrent layers. The authors claimed that QANet has the advantage of faster training because it does not have long dependency chains in its computation graphs which makes it easy to parallelize on GPUs. With the improvement in training time, they were able to train the model on a larger augmented dataset and out-performed other models. We follow the detailed description in the paper and re-implemented the model ourselves.

1. **Input Embedding Layer**: We use the same input embedding layer with character embedding as BiDAF.

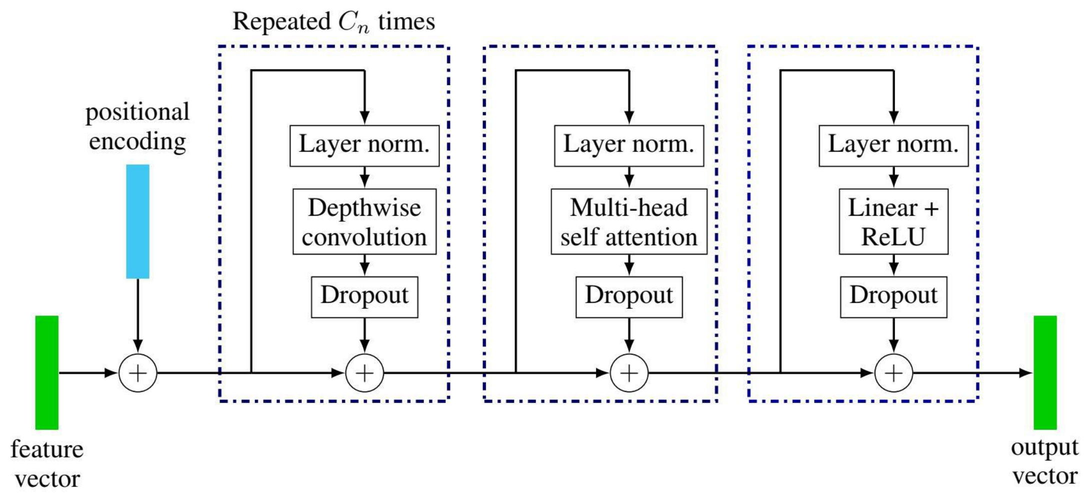

Figure 2: The structure of the encoder block in QANet. The convolution block is repeated and cascaded C_n times. The blue dash blocks are the residual blocks where we apply layer dropout.

2. **Embedding Encoding Layer**: The embedding encoder layer is a single "encoder block", which an Encoder block is a structure we will also use in later layers. Figure 2 shows the structure of an encoder block. First, we apply positional encoding to the input. We used the same sinusoidal-based encoding as in Transformer [5]. Then, the input will go through C_n convolutional block. In each convolutional block, we apply layer nomalization [11], depthwise separable convolutions [12], and dropout to the input vector. The convolutional block is treated as a residual block, so the output from the convolutional block is summed with its input, and the result is passed to the next layer. Similarly, the feature vector will then go through multi-head self-attention and a feed-forward block, and these blocks are also treated as residual blocks. We apply layer dropout [13] on all the residual blocks.
3. **Context-Query Attention Layer**: Although the authors of QANet suggested using DCN attention, we used the context-query attention layer from BiDAF directly.
4. **Model Encoding Layer**: The model encoder layer consists of three sub-layers. Each sub-layer is formed by stacking multiple encoder blocks, and these three sub-layers are also cascaded together. The weights are shared between these three sub-layers. As in Figure 3, let o_1, o_2, and o_3 be the output from the first, second, and third sub-layer. These three vectors combined are the output of this layer, and we feed them to the next layer.
5. **Output Layer**: The output layers are given o_1, o_2 and o_3. We concatenated these vectors and obtain $\boldsymbol{o}_s = [o_1; o_2]$ and $\boldsymbol{o}_e = [o_2; o_3]$. $\boldsymbol{o}_s$ and $\boldsymbol{o}_e$ are then passed to two feed-forward layers with separated weight to obtain $\boldsymbol{p}_s$ and $\boldsymbol{p}_e$, the log-probability prediction vectors.

4.2 Data Augmentation

We used back translation to augment the training data. The goal is to translate texts in the training data to a chosen language and then translate them back to English. By doing so, we get additional texts with similar meanings and wording slightly different than the original one, which we can incorporate in our training dataset.

To do the translation, we used the fairseq [14] toolkit. The specific model we used was a Transformer model [15] that won Facebook FAIR's WMT19 competition. We chose Germany as the intermediate language as it has a structure similar to English. It is the only language with both forward and backward translating available in fairseq toolkit.

With this translation model, we did back translation on the reading passages and the questions. For the reading passages, the procedure is more complicated because back translation might change the position of the span of the correct answer. Instead of trying to recover the answer span, we avoid the problem simply by not doing back translation on sentences that overlap with the answer

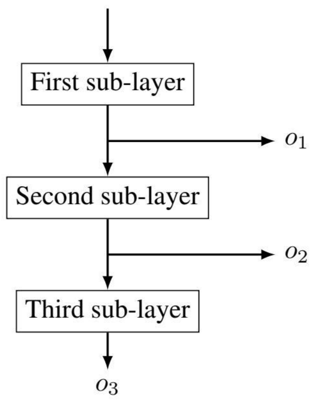

Figure 3: The structure of the model encoder block in QANet.

span. Specifically, let k be the number of sentences in the passage. If the answer span lies in the i-th sentences (if the answer span across two or more sentences, we ignore the question) where $i \leq \lfloor N/2 \rfloor$, we only translate sentence $\lfloor N/2 \rfloor + 1, \lfloor N/2 \rfloor + 2, \ldots, N$. In this way, we will never modify the sentence that contains the answer.

4.3 Ensemble Modeling

We used ensemble methods to combined the result from different models. We experiemented the following ensemble methods:

1. **Weighted average**: Let $\boldsymbol{p}_{i,s}$ and $\boldsymbol{p}_{i,e}$ be the log-probability vectors of each position being the start and the end of the span output by the i-th model. We assign a weight w_i to each model. These weights are set based on their performance on the dev set. Then, we computed the weighted average of the log-probability vectors by $\overline{w} = \sum w_i p_{i,s}$. Finally, we use this log-probability vector to compute the optimal start and end of the span, as we did in a single model.

2. **Majority voting**: Majority voting: Similarly, we assign a weight w_i to each model. Then we collect the output span (s_i, e_i) from all models to form a list of output candidates and sum up the model weights for each of the spans. We output the span with the largest weight sum.

5 Experiments

5.1 Data

The task is question answering, and the dataset we used is a modified version of the SQuAD 2.0 dataset [16]. SQuAD's training dataset has a total of 129941 context-question pairs. Each pair consists of a paragraph and a question about the paragraph. Our models will choose a span of the text as the answer to the question or output that the answer cannot be found in the paragraph.

5.2 Evaluation method

For our evaluation, we use two metrics, which are **Exact Match (EM)** score and **F1** score. EM is 1 if the model output exactly the same span as one ground truth answer, and 0 otherwise. The F1 score is calculated on the words in the answer span selected by the human and the model. We focused on the F1 score when evaluation results between models, as EM score does not differentiate an answer that is only one word off the reference solution and an answer that is not related to the solution.

5.3 Experimental details

5.3.1 BiDAF Improvement

After adding the character embeddings to the baseline BiDAF, we trained the model using the same model and training parameters in the default code. Specifically, the hidden size is 100, and the dropout probability is 0.1.

5.3.2 QANet Implementation

We followed the description of QANet in the original paper [2] to re-implement the model. We experimented with different parameters and compared their performance. In all experiments, we use the ADAM optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. We set the learning rate to increase inverse exponentially from 0.0 to 0.001 in the first 1000 steps and then remain constant throughout the training. We set the dropout probability of 0.05 for the character embedding layers and 0.1 for all other dropout layers. We also applied layer dropout to residual blocks in each embedding/model encoder layer such that the entire l-th residual blocks will be dropped with probability $0.1(l/L)$, where L is the total number of residual blocks in the layer. The original paper suggested these parameters. Table 1 listed parameters that were changed in each experiment.

Model	(1)	(2)	(3)	(4)	(5)
Hidden Size	128	128	128	256	64
Heads in Self-Attention Layer	2	4	8	8	12
Encoder Blocks in the Model Encoder Layer	3	3	3	3	2

Table 1: Parameters of different QANet experiments

5.3.3 Character embedding

We compare the results after removing the character embedding layer from QANet. All other hyperparameters are fixed.

5.3.4 Data augmentation

We used the method described in the previous section to create an additional training set with 79674 questions. We combined it with the original training dataset to form an augmented dataset of 209615 questions, which is an increase of 61%. The data augmentation step took a substantial amount of time because the translation model was computational heavy, and this was the largest augmented data we could get before the deadline. We then trained BiDAF and QANet on this new dataset and compare the performance when using the original dataset.

5.3.5 Ensemble Modeling

We used a total of 5 QANets we trained to perform a model ensemble. These QANets were trained with different parameters, which provided some diversities. We chose not to use BiDAF models because there is a large gap between the score of BiDAF and QANet, and based on our experiment, adding them to the ensemble model did not increase the F1 score, if not made it worse. We used both the weighted average and majority voting and tried equal weights and weights based on their performance.

5.4 Results

Table 2 summarizes the results of our models on dev dataset. After adding character embeddings to BiDAF, it successfully improves the F1/EM score by 3.10/2.85.

Overall, our QANet models outperform BiDAF models. In QANet (1), (2), and (3), we found that the performance improved if we increase the number of heads in self-attention layers. For QANet (3) and (4), we fixed the number of heads in the self-attention mechanism and doubled the hidden dimension in (4). Yet, we got similar results from the two models. For QANet (5), we originally predicted we could improve the score again by increasing the number of heads in self-attention layers.

However, we had to reduce the hidden dimension to 64 due to GPU memory limitation, and we got results similar to QANet (3) and (4).

In the data augmentation experiment, we found out that our data augmentation method does not help at all. When training the BiDAF model with augmented data, it slightly improved the scores. However, the difference is very small that falls into the error range.

Finally, the experiment shows that ensemble modeling can improve performance. The best ensemble model improves the F1/EM score by 2.34/2.64 and 3.17/3.67 in the dev and test dataset.

Model	F1	EM	AvNA	Test F1	Test EM
BiDAF	60.84	57.69	67.94	-	-
BiDAF + Char. Embeddings	**63.94**	**60.60**	**70.01**	-	-
QANet (1)	64.34	60.54	71.42	-	-
QANet (2)	67.56	63.92	73.79	-	-
QANet (3)	**68.49**	**64.53**	**75.16**	**65.54**	**61.71**
QANet (4)	68.27	64.49	74.71	-	-
QANet (5)	68.43	64.29	74.84	-	-
QANet w/o Char. Embeddings	65.04	61.86	71.13	-	-
BiDAF + Char. + Data Aug.	**64.03**	**60.73**	70.22	-	-
QANet(2) + Data Augmentation	67.52	63.85	74.24	-	-
Ensemble (Weighted Average)	**70.83**	66.96	**76.39**	68.40	64.93
Ensemble (Majority Voting)	70.47	**67.17**	76.09	**68.71**	**65.38**

Table 2: The performance of each model

6 Analysis

6.1 QANet

We give an analysis of the QANet models we trained, focusing on the training data that they gave wrong answers to. In the following, we inspected the output from our best single model QANet(3).

In Table 4, we selected three incorrect answers produced by our model. We made ground truth spans in the table with bold font and summarize our model's deficiencies using the skills proposed in [17]. We can find that our model is not good at causal relation, logical reasoning, and co-reference.

The first example demonstrates that our model is not able to capture causality in context. Northern Chinese ranked high is because they surrendered to the Mongols, but our model cannot capture this causal relation to predicting the answer.

The second example shows that our model does not perform well on logical reasoning. In this context, we find that *"Scotland married Edgar's sister Margaret"*, which implies that Edgar is Margaret's brother. However, our model does not get this information to make the answer.

The last example indicates that our model does not capture co-reference well in context. The word *force* in the first sentence: "This means that the unbalanced centripetal **force** felt by any object is always directed toward the center of the curving path", and in the second sentence: "Such **forces** acts perpendicular to the velocity vector associated with the motion of an object," are the same. However, our model does not learn this concept and generates the wrong answer.

Inspired by [18], we produced Table 3 showing the performance of QANet(3) on different types of questions (What, Who, How, etc.).

In table 3, we can find that our model performs best on *When* type questions. The reason can be that the answers may include numbers that make our model easier to retrieve. Interestingly, our model performs second-best on *Which* type questions. We find that 42 out of 214 questions ask about "Which country" that make model focuses on country names in contexts and performs well on this type of problem. On the other hand, our model performs worst on *Why* type questions. This is because these questions need more logical reasoning and inference during prediction.

Type	Overall	What	Who	How	When	Where	Which	Why	Other
Count	5951	3567	627	571	455	256	214	86	65
F1	68.49	67.99	70.86	64.42	**74.79**	65.73	74.26	60.60	39.43
EM	64.53	63.73	68.71	59.63	**72.79**	61.03	70.33	52.27	33.90
AvNA	75.16	75.23	74.43	69.85	**79.82**	73.16	81.30	73.86	54.28

Table 3: Query Types and Performance

6.2 Data Augmentation

We are really curious why our data augmentation method did not work out. One possibility is that the translation quality was poor, causing the back-translated text to be dramatically different from the original text in style. This causes the model to overfit in detecting this difference to locate the position of the answer span. Thus, we do a qualitative analysis and check if the augmented data is starkly different from the original one.

An back-translated context and questions are shown in Table 5. We can see that the translation quality is acceptable. We suspected that because the translation model we used gives precise translation, especially for the reading passage, it provided nearly no changes and diversity to the text, which render the method ineffective.

We can see that the translation quality are quite good. In fact, we suspect that because the translation model we use give precise translation, it provides to less changes and diversity to the text, which render the method ineffective.

7 Conclusion

In this project, we re-implemented the QANet model that achieved an F1/EM score of 68.49/64.53 and 65.54/61.71 in the dev and test dataset, respectively. We tested the QANet with different parameters and compared their performance. We also showed that character-level embedding is crucial in this task. After we added the character embedding layer to BiDAF, we successfully improved the scores of the model. We also did an experiment showing that when the character embedding layer is removed from the QANet model, the performance dropped significantly.

We tried to do data augmentation on the training dataset by using back-translation. Surprisingly, the models training on the augmented dataset did not show any improvement. We hypothesized that the reason is that our back-translation method or model did not provide enough diversity to the training data. Further experiments are needed to prove our assumption.

Finally, we used the ensemble method to further boosted our models. In the end, the ensemble model achieves an F1/EM score of 70.47/67.17 in the dev dataset and 68.71/65.38 in the test dataset, which is the best score we have achieved.

References

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

[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. *arXiv preprint arXiv:1804.09541*, 2018.

[3] Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. May 2017.

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

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Context	Question	Prediction	Lack of Skills
The reason for the order of the classes and the reason why people were placed in a certain class was the date they surrendered to the Mongols, and had nothing to do with their ethnicity. **The earlier they surrendered to the Mongols, the higher they were placed**, the more the held out, the lower they were ranked. The Northern Chinese were ranked higher and Southern Chinese were ranked lower because southern China withstood and fought to the last before caving in.	Why were Northern Chinese ranked higher?	N/A	Causal Relation
One of the claimants of the English throne opposing William the Conqueror, **Edgar Atheling**, eventually fled to Scotland. King Malcolm III of Scotland married Edgar's sister Margaret, and came into opposition to William who had already disputed Scotland's southern borders. William invaded Scotland in 1072, riding as far as Abernethy where he met up with his fleet of ships. Malcolm submitted, paid homage to William and surrendered his son Duncan as a hostage, beginning a series of arguments as to whether the Scottish Crown owed allegiance to the King of England.	Who was Margaret's brother?	N/A	Logical Reasoning
This means that the unbalanced centripetal force felt by any object is always directed toward the center of the curving path. Such forces act **perpendicular** to the velocity vector associated with the motion of an object, and therefore do not change the speed of the object (magnitude of the velocity), but only the direction of the velocity vector.	How do centripetal forces act in relation to vectors of velocity?	N/A	Co-Reference

Table 4: Examples of Errors in Predictions of Answerable Questions

Original Text	Back-translated Text
A global power city, New York exerts a significant impact upon commerce, finance, media, art, fashion, research, technology, education, and entertainment, its fast pace defining the term New York minute. Home to the headquarters of the United Nations, New York is an important center for international diplomacy and has been described as the cultural and financial capital of the world.	A global power city, New York exerts a significant impact upon commerce, finance, media, art, fashion, research, technology, education, and entertainment, its fast pace defining the term New York minute. Home to the headquarters of the United Nations, New York is an important center for international diplomacy and has been described as the cultural and financial capital of the world.
What American city welcomes the largest number of legal immigrants?	Which American city is home to the most legal immigrants?
What city has been called the cultural capital of the world?	Which city has been designated as the World Capital of Culture?

Table 5: Examples of back-translated text

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942, 2019.

[7] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics.

[8] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[10] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks, 2015.

[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[12] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.

[13] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016.

[14] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[15] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fair's WMT19 news translation task submission. *CoRR*, abs/1907.06616, 2019.

[16] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[17] Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 806–817, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[18] Pablo Veyrat Guillaume Nervo. Extended qanet on squad 2.0. 2020.