Building a QA system (Default IID SQuAD track)

Stanford CS224N Default Project

TA Mentor: Kathy Yu

Aaron Li
Department of Computer Science
Stanford University
aaronli9@stanford.edu

Edgar Roman
Department of Computer Science
Stanford University
emroman@stanford.edu

Tanish Jain
Department of Computer Science
Stanford University
tanishj@stanford.edu

1 Abstract.

In this project, we aim to develop a more complex model from a starting baseline model, namely BiDAF without character embeddings, to perform well on the SQuAD 2.0 dataset, which is relevant to evaluating the model's text-comprehension ability. We ultimately find that the integration of character embeddings into the baseline model enhances performance as does the use of the Dynamic Coattention Network on "answerable" questions.

2 Introduction.

In this paper, we implement various question-answering models and evaluate their performance on the second version of the Stanford Question Answering Dataset or SQuAD2.0. This dataset extends its prequel with the addition of "unanswerable" questions, discussed further in Section 3.

We look at three architectures primarily. The first model is based on Bidirectional Attention Flow (BiDAF) [1]. The second model relies on the same architecture but expands it by using character embeddings. The third model is based on the Dynamic Coattention Network [2]. Additionally, we also implement a transformers-based model, although we were unable to complete training the model due to computational resource constraints.

The first model, based on BiDAF, serves as the baseline. We compare the performance of each of these models on SQuAD2.0 and evaluate their ability to improve on the baseline performance. This is relevant to evaluating the text-comprehension ability of any model. In addition, it has several practical applications, such as answering questions by humans using information from large pieces of text, as in the case of Google's question answering system.

3 Related Work.

In this project, we use the Stanford Question Answering Dataset version v2.0, or SQuAD2.0. The first version of this dataset consists of over 100,000 questions created by crowdworkers on Wikipedia articles. The answer to each question corresponds to an exact segment of text from an associated passage from the article (the "context") [3]. SQuAD2.0 combines this dataset with "unanswerable" questions written by crowdworkers to look similar to answerable ones. [4]. This means that for any question answering model to perform successfully on this dataset, it must first determine if the question is answerable at all, and if so, answer it appropriately. Rajpurkar et al., therefore, present this as a dataset that may be useful for developing more robust NLP models that may be useful in more practical contexts where no a priori assumption is made about the answerability of questions [4].

Specifically, the dataset consists of question, answer and context triples, where the answer of a question is a segment of the context. An example of this is shown below.

Question: What is the name of the state that the megaregion expands to in the east?

Context: The 8- and 10-county definitions are not used for the greater Southern California Megaregion, one of the 11 megaregions of the United States. The megaregion's area is more expansive, extending east into Las Vegas, **Nevada**, and south across the Mexican border into Tijuana.

Answer: Nevada

Source: Rajpurkar et al. [4]

Our first two model architectures are based on Bidirectional Attention Flow. The baseline model uses word-level and contextual embeddings and utilizes bi-directional attention flow "to obtain a query-aware context representation" [1]. The second model is based on Seo et al.'s work and also includes character-level embeddings [1]. While Seo et al.'s work is based on SQuAD, we re-implement their approach to assess its applicability on SQuAD2.0.

The third model architecture relies on Dynamic Coattention Networks [2]. This approach also utilizes bidirectional attention. A key feature of this architecture is the dynamic pointing decoder which can dynamically change the pointers demarcating the answer span.

The final model architecture is the Transformer-based model QANet. QANet modifies the traditional Transformer architecture by replacing RNNs with self-attention and convolution. Specifically, it uses "depthwise-separable convolution to capture local dependencies in the input sequence" for its encoder block [5]. With this, it is able to achieve high levels of performance on question-answering tasks, relevant for this project.

4 Approach.

For our approach, we test out a few different techniques from prior work mentioned in Section 3. Each technique is discussed below. We then compare and analyze their performance.

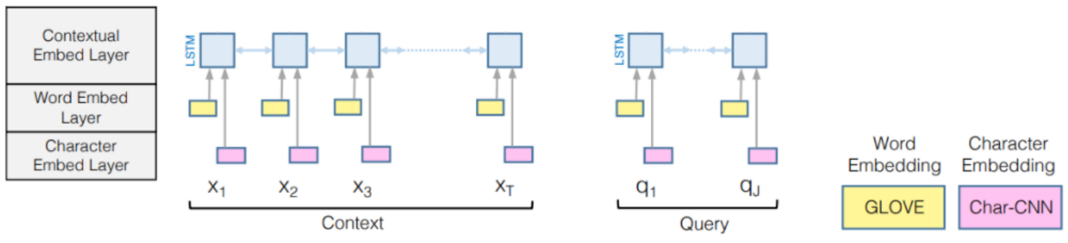
4.1 Baseline

Our baseline model is based on Bidirectional Attention Flow (BiDAF) [1]. However, unlike the original model, the implementation of this baseline model does not include a character-level embedding layer. Nonetheless, since our baseline model is largely based on the original BiDAF model, we will not be describing it in too much detail, and we will instead provide the original paper for reference.

4.2 BiDAF with Character Embeddings

It appeared that the baseline model performed relatively well on its own. However, we were interested in exploring whether we could modify the baseline model, which we know is based on BiDAF, by extending it to match the original BiDAF-No-Answer model [] by introducing character-level word embeddings in addition to the word-level embeddings, and discovering whether or not this would improve the baseline model's performance. The idea behind implementing character-level embeddings in addition to the word-embeddings is that we wanted to be able to condition on the internal structure of words. Furthermore, we wanted to be able to better handle out-of-vocabulary words (OOV). Instead of simply allowing the model to deal with OOV words by having GloVe assign them a random embedding, character-level embeddings could potentially allow us to find better embeddings by focusing on the character-level compositions of these OOV words. Lastly, we also know that for our evaluation metrics, we are using Exact Match (EM) and F1 score, which measures the weighted average of the precision and recall rate at the character level.

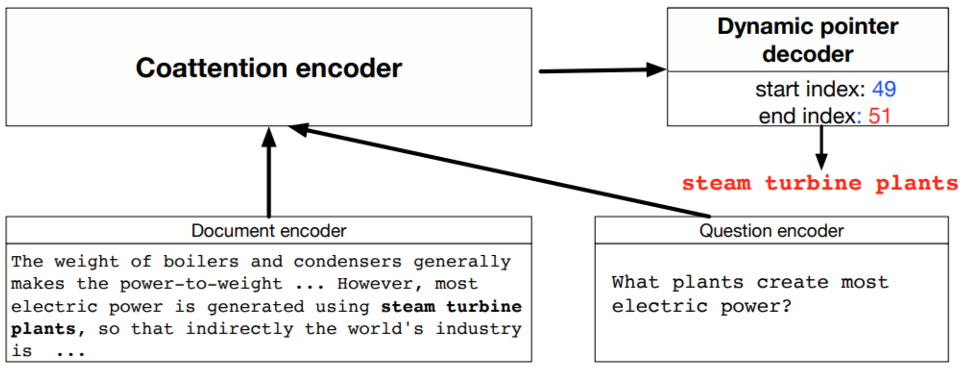
To implement character embeddings, we added a character embedding layer that utilizes a simple Convolutional Neural Network (CNN), whose output is then max-pooled in order to obtain a fixed-size vector for each word. To integrate the embeddings into our model, we later concatenate both the character and word embeddings, which is ultimately passed into a two-layer Highway Network [6] just as in the original BiDAF model.



BiDAF Model
Source: Seo et al. [1]

4.3 DCN

We implemented a Dynamic Coattention Network based on [2]. This approach was picked for its ability to use bidirectional attention, much like the first approach, alongside the introduction of a dynamic pointing decoder that goes over answer spans. This can be particularly useful, since this allows the model to "explore local maxima corresponding to multiple plausible answers" [2]. The decoder can then produce start and end indices of the answer, as shown in Figure **??**.



```
The weight of boilers and condensers generally
makes the power-to-weight ... However, most
electric power is generated using steam turbine
plants, so that indirectly the world's industry
is ...
```

```
What plants create most
electric power?
```

Overview of DCN Encoder-Decoder Setup
Source: Xiong, Zhong and Socher. [2]

The implementation was built on top of the provided baseline implementation, with a few key modifications.

- **Addition of coattention layer.** The attention layer of the baseline model was modified into a coattention layer. Using the strategy described in the project handout, we compute the Context-to-Question (C2Q) Attention outputs and the Q2C attention outputs. The latter are concatenated with the former, and passed to a bidirectional LSTM, which completes the coattention layer. Specifically, this may be represented as

$$\{u_1, ..., u_N\} = biLSTM\{[s_1, a_1], ..., [s_N, a_N]\}$$

  where $u_1, ..., u_N$ are the resulting hidden states (i.e., the coattention encoding), a_i are the C2Q attention outputs, and s_i are the weighted sums of the Q2C attention outputs for $i \in 1, ..., N$.

- **Addition of a dynamic pointing decoder.** Another key feature of the DCN is the dynamic pointing decoder. At each iteration, the decoder state is updated using previous estimates of the start and end positions to generate the new estimates of the start and end positions, as follows:

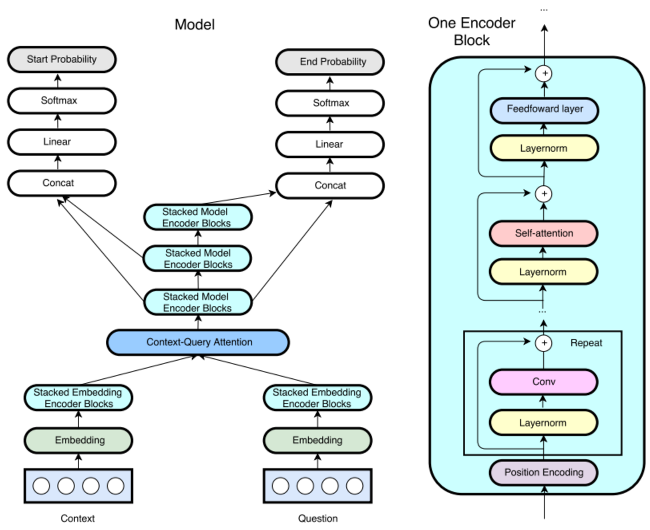


Figure 1: QANet model architecture (left) and single encoder block (right). *Source: Yu et al. [5]*

$$h_i = LSTM_{decoder}(h_{i1}, [u_{s_{i-1}}; u_{e_{i-1}}])$$

where h_i, s_i, and e_i are the hidden state, the estimate of the start position, and the estimate of the end position for iteration i.

Since the original paper did not list all relevant hyperparameters, we relied on the implementation by André Jonasson [7], which exhibits similar levels of performance. We made the simplifying assumption that these hyperparameters would be suitable, at least as a starting point, for our model, although the implementation differs slightly.

4.4 QANet

The transformer-inspired architecture was based on QANet, introduced by Yu et al. [5]. QANet improves on prior models by speeding up training and inference while still providing comparable to better performance. A key reason for this is it does away with the recurrent architecture.

The model consists of 5 layers: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer and an output layer. Importantly, for both the encoder layers (i.e., the "Encoder Block"), it uses convolutional and self-attention mechanisms instead of RNNs [5]. The model architecture and the Encoder Block are shown in Figure 1.

The QANet implementation draws from some of the prior works mentioned in Section 3. For instance, the Context-Query Attention Layer utilizes DCN attention for its query-to-context attention mechanism [5]. While Yu et al. perform data augmentation, we do not make any changes to the data. This is crucial to ensure a fair comparison across models.

5 Experiments.

5.1 Data

The evaluation was done using the SQuAD 2.0 dataset, which consists of triples containing questions, contexts and answers [4]. Specifically, the models were evaluated in the dev set in the provided data splits.

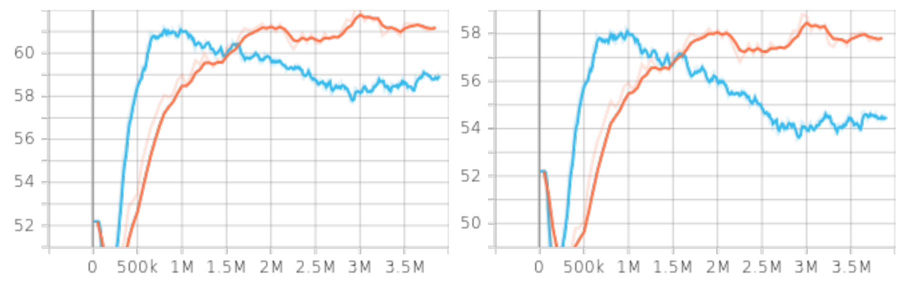


Figure 2: F1 & EM Scores For BiDAF w/ Hyperparamter Tuning

5.2 Evaluation Method

The evaluation was done using the F1 and EM scores, which have been discussed in the project handout and project proposal.

The baseline model with the character embeddings was run using the hyperparameters in Table 1. The training time was approximately three hours.

Hyperparameter	Value
Learning Rate	0.5
Size of Char Embedding	1376
Size of Hidden Units	100
Dropout	0.2
Optimizer	Adam
Batch Size	64

Table 1: BiDAF (baseline + character embeddings) Hyperparameters

In addition to training the BiDAF model (baseline + character embeddings) with the above hyper-parameters, we also attempted to run the same model but with a reduced number of steps between successive evaluations as shown in Figure 2. By default, the weights of the model were evaluated every 50000 steps (orange) but we reduced this number to every 10000 steps (blue). Furthermore, in addition to reducing the number of steps between successive evaluations, we also reduced the dropout probability to 0. The idea behind these changes was to increase the speed of training since training time took approximately three hours. What we found was that it indeed took the model with a lower number of steps between successive evaluations fewer steps (about a third) to converge to similar “best weights”. We would like to note, however, that although we were able to reduce the training time, this came at a cost: a slight decrease in the model’s performance on the dev set. This is most likely due to our lowering of the dropout probability to 0, which meant that we were no longer employing a regularization technique to reduce over-fitting and improve generalization error. Nonetheless, the difference in performance was only marginal while the reduction in training time was relatively significant.

The Dynamic Coattention Network was run using the hyperparameters in Table 2. This was based on implementations discussed in [2] and [7].

Hyperparameter	Value
Learning Rate	0.001
Size of Embedding	100
Size of Hidden Units	100
Dropout	0.3
Optimizer	Adam
Batch Size	64

Table 2: DCN Hyperparameters

The final QANet model was run using the hyperparameters in Table 3. This was based on implementations discussed in [5].

Hyperparameter	Value
Learning Rate	0.001
Size of Embedding	200
Size of Hidden Units	128
Dropout	0.2
Optimizer	Adam
Batch Size	60

Table 3: QANet Hyperparameters

Due to computational resource limitations, we first decided to locally run the model with a small batch size of 32. However, the model performance did not significantly improve, and consistently remained below the baseline. We, therefore, trained the model with the recommended batch size of 60 subsequently, but due to the aforementioned constraints, were not able to train it for an extended period of time. Nevertheless, due to the speedup introduced by the model, we were still able to generate significant results despite the relatively short training time.

5.3 Results

The results of each of the models are summarized in Table 4.

Model	F1 Score	EM Score
Baseline (BiDAF w/o Character Embeddings)	61.769	58.629
BiDAF	62.107 (+0.338)	58.814 (+0.185)
Dynamic Coattention Network	58.704 (-3.065)	57.269 (-1.361)
QANet	55.800 (-5.969)	52.810 (-5.819)

Table 4: Summary of model performance

6 Analysis.

As can be seen in Table 4, the BiDaF model, which is essentially the baseline model with the addition of character embeddings, performs marginally better than the baseline model (+0.338 for F1 and +0.185 for EM), which is expected. However, we believed that the improvement would be slightly more substantial. For this reason, perhaps it would be beneficial to look into pretrained character embeddings as opposed to randomly initialized character embeddings that are improved over time. In addition, since we noticed a slight improvement in performance by using these embeddings, we are also interested in the possibility of making use of other kinds of subword modeling.

As noted in Table 4, both the F1 and EM scores for the Dynamic Coattention Network model were below the baseline. This is worse than expected since the addition of the coattention layer and dynamic pointer were seen as improvements in [2]. Although it is unclear why the model did not perform as expected, it may be because of the choice of hyperparameters, including the training time. The model was unable to complete training due to resource constraints, and the predictions were made using an intermediate model, which may underline the worse-than-expected results. Interestingly, most of the prediction error appears in the "unanswerable" questions, where the model performs poorly. An example of this is shown below.

Question: What is France a region of?

Context: The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia. Through generations of assimilation and mixing with the native Frankish and Roman-Gaulish populations, their descendants would gradually merge with the Carolingian-based cultures of West Francia. The distinct cultural and ethnic identity of the Normans emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries.

True Answer: <No Answer>

Predicted Answer: France

This may be indicative of the inability of the model to generalize well to unanswerable questions, especially since the original model in [2] was developed using the SQuAD dataset, which only contained answerable questions.

We implemented QANet to address this issue. Since QANet incorporates important characteristics of both BiDAF and DCN and has shown improved performance on SQuAD [5], it promises to replicate this success on SQuAD2.0. While QANet performed worse than the other models, the results more consistent than those of the DCN model. First, the error was more evenly split between answerable and unanswerable questions. Second, the model consistently improved its performance during training. The model training was timed out, and therefore it is difficult to evaluate the true performance of the model.

7 Conclusion.

This paper set out to develop a more complex model compared to the starting BiDAF baseline model without character embeddings in order to perform well on the SQuAD 2.0 dataset. While it is certainly is the case that we managed to develop a more complex model, it was not necessarily the case that the model performed significantly better or as well as we would have hoped. Nonetheless, we gained insights and a deeper understanding of how the integration of character embeddings enhances the performance of the model as well as how a Dynamic Coattention Network performs more strongly on “answerable questions”. We also discovered, more recently, that it is possible to train the BiDAF model much more quickly than the original 3+ hours, which would enable us to try out various other methods. Similarly, the QANet model was found to produce relatively good results in a short period of training. As a result, we believe that we would be able to produce a more strongly performing model given more time and potentially more computational resources. This being said, that does not take away from our experience working on this project and the knowledge we gained by doing so.

8 Future work.

As mentioned earlier, we found that the main shortcoming of dynamic co-attention is poor performance on the unanswerable questions within the SQuAD 2.0 dataset. The SQuAD 2.0 dataset specifically contains adversarially generated unanswerable questions that look like they could be answered by the text, but actually are not [4]. Crowdsourced workers provided "plausible answers" to unanswerable questions, and it was found that about half the incorrect answers to unanswerable questions provided by previous models closely matched these plausible answers [4]. Rajpurkar, Jia, and Liang further found that when using automatically generated unanswerable questions, rather than adversarially generated ones, model performance was not significantly hindered [4]. Such findings indicate that it’s not the fact that a question is unanswerable, but rather the seeming relatedness of the question to the provided text that fools dynamic co-attention. If a question is guaranteed to be answerable, then our dynamic co-attention model does well because it can identify similar words within the passage that help answer the question. Thus, our dynamic co-attention model is pointing to sections of text that are likely to be an answer based on how similar the words are to each other in the question and the text. However, our model doesn’t consider that semantically, a selected span of words being similar doesn't mean that span provides an answer to the question; in fact, the same words could be used in different ways that people pick up on easily but that are difficult for a Neural Network to represent, as seen in the above example when our model answered that

Our group had planned to implement an approach called "Hermitian Co-Attention Networks for Text Matching in Asymmetrical Domains" [6], as described by Yi Tay, Anh Tuan Luu, and Siu Cheung Hu. By taking advantage of inner products in the complex space [6], Hermitian Co-attention can more successfully represent nuanced Unfortunately, we were unable to achieve working code for Hermitian Co-attention, but this is something we believe could help.

Aside from a new model, some further hyperparameter tuning could serve us well. For example, due to time constraints with training, our dropout for training embeddings was 0, which we believe is not optimal because it interferes with generalization.

References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2018.

[2] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering, 2018.

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

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

[5] 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.

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

[7] André Jonasson. Dynamic coattention network plus - question answering, 2018.