Machine Reading Comprehension with Abstention Verification

Stanford CS224N Default Project

Mara Finkelstein
Department of Computer Science
Stanford University
mfinkels@stanford.edu

Abhishek Rawat
SCPD Student
Stanford University
arawat@stanford.edu

Rakesh Grewal
SCPD Student
Stanford University
rgrewal@stanford.edu

Abstract

In this project, we aim to build a robust question answering system using the SQuAD 2.0 dataset for training and evaluation. The primary focus of our project is to explore non-recurrent solutions for SQuAD built off of the QANet architecture. We implemented the QANet model from the ground up with emphasis on modular implementation and computational efficiency. We further enhanced it with an abstention module and variable-sized filters in convolutional layers. Additionally, we trained various Reader and Verifier models and integrated those to achieve better performance metrics. We used EM and F1 scores, as well as answer versus no-answer classification accuracy, to evaluate the performance of our model. Our custom QANet-based models achieved well-placed scores on the non-PCE leaderboard.

1 Introduction

Machine reading comprehension is a classic natural language processing task that has received much attention since 2016 with the release of the SQuAD dataset [1] and the launch of the accompanying competition. The task for the SQuAD competition is extractive question answering, in which a query and context (generally a single paragraph) is provided, and the exact answer to the query is present in the context. In the SQuAD 2.0 dataset, there are also some contexts for which no answer to the query is present, and in this case, the model should decide to abstain. We compete in the SQuAD 2.0 non-PCE division, and our investigation centers around two themes: First, we experiment with non-recurrent neural architectures for the question answering task and second, we investigate the limitations and benefits of multi-task versus single-task modeling in the context of predicting both answer spans and classifying whether or not an answer is present.

While recurrent neural networks (RNNs) naturally suit the sequential nature of language, they require that tokens be fed into the model in order, which prevents parallel computations. Convolutional neural networks (CNNs), on the other hand, use a sliding filter of shared parameters to capture local context in the input, and are easily parallelizable since each sliding window computation is "memoryless," with no knowledge of previous hidden states in the sentence or corpus. While CNNs have historically been used to capture spatial, rather than sequential, structure in data, [2] proposes a novel non-recurrent reading comprehension model, called QANet. We re-implement the QANet model in this paper, while exploring modifications and augmentations.

The QANet model is not specifically designed to handle "answer-abstention" samples of the SQuAD 2.0 dataset, in which some of the questions do not have an answer in the given context. Proper handling of this category of samples requires that a model perform language inference more precisely. Specifically, the model would need to excel at the tasks of determining entailment, contradiction, or neutrality between queries and proposed answers [3]. We experiment with
abstention decoder modules trained to perform inference modeling, and show that the use of these modules on top of a base reader can improve performance.

2 Related Work

The most successful (non-PCE) non-recurrent model to date is QANet [2], which offers two primary contributions to the field of machine reading comprehension: a faster, and still highly performant, model with no recurrent networks, and the successful use of data augmentation to boost model performance. Speed confers several advantages, including non-prohibitive model runtimes when scaling up to process longer pieces of text (especially useful in industrial NLP applications), and allowing for a larger training dataset due to faster per-example training time. The authors investigate this second possibility by augmenting the existing training set with additional examples, through the use of backtranslation.

A key challenge in the SQuAD 2.0 competition is the multi-task objective of answer extraction and no-answer detection. The QANet model achieves less than 75% accuracy in classifying answerable versus unanswerable questions, which leaves much room for improvement. Several recent approaches have attempted to improve no-answer detection, including the Read+Verify [3], U-Net [4], Retrospective Reader [5], and NeurQuRI [6] models. The Read+Verify model adopts a Reader module with independent span and no-answer losses in addition to a joint loss, supplemented by a separate Verifier model, which decides whether the predicted answer is entailed by the query, thereby validating the answerability of the question. The U-Net model resembles Read+Verify, with the main difference being that the model is trained end-to-end using a multi-task framework. The U-Net model also introduces a “universal node”, which acts on both the question and passage to learn whether the question is answerable. The Retrospective Reader is trained end-to-end as well and, rather than conforming to a verifier framework as in the previous two models, uses “sketchy” and “intensive” reader modules which are trained in parallel, and whose results are concatenated to produce a final prediction. In our project, we build a variant of the Read+Verify model.

3 Approach

We experimented with RNN and CNN-based model architectures for reading comprehension, and with both model augmentation and pipelined models for abstention detection.

3.1 BiDAF Baseline

For our first baseline, we used the provided BiDAF [7] implementation, which consists of a word embedding layer, a RNN encoder, a context-query attention layer (with bi-directional attention flow), a RNN model encoder layer, and a final output layer consisting of two fully-connected + softmax heads which predict start and end positions of the answer. In order to enhance the baseline model, we added character embeddings, as in the original BiDAF implementation [7]. In particular, we used pre-trained character vectors, which we passed through a one-dimensional CNN followed by max-pooling to obtain word-level representations. We then concatenated the character and word embeddings, before passing them to a two-layer Highway Network [8].

3.2 Custom QANet Implementation

The QANet model consists of the following five modules: embedding layer, encoder layer, context-query attention layer, model encoder layer, and output layer. First, input embeddings for the query and context are obtained from the concatenation of pre-trained GloVe vectors [9] and convolutional outputs of character embeddings, as in [10]. The context and query embeddings, in addition to their positional encoding, as in [11], are fed into two encoder blocks, which are used to separately encode the query and context. The QANet model forgoes all recurrence and instead proposes an encoder block which combines depthwise separable convolutions with multi-headed self-attention (using a simple dot product attention function). The conceptual rationale behind the QANet encoder block is that convolutions capture local structure in the input, while self-attention captures global structure. A final dense layer is stacked on top of the convolutional and self-attention layers in the encoder block, and each of these layers f (convolutional, self-attention, and dense) are
placed inside a residual block with layer normalization [12]. In particular, for each layer $f$ and input $x$, the output of the layer is $f(\text{layernorm}(x)) + x$.

The encoder layer then outputs a context representation $C \in \mathbb{R}^{d \times n}$ and a query representation $Q \in \mathbb{R}^{d \times m}$, where $d = \#$ of convolutional filters, $n = \#$ of words in context, and $m = \#$ of words in query. $C$ and $Q$ are then passed jointly to a context-query attention layer, which is based on the BiDAF model’s attention flow layer [7] and the Dynamic Coattention Network’s coattention encoder [13]. The concatenation of the contextual embeddings (from the embedding encoder layer) and attention vectors (from the context-query attention layer) are then passed into three more QANet model encoder blocks with shared weights. The first and second of these blocks is concatenated, then passed through linear and softmax layers to predict the start position of the answer, while the first and third of these blocks is used to predict the answer’s end position.

Our QANet implementation includes trainable character embeddings, frozen GloVe word embeddings, and positional encodings, which are fed into stacked encoder blocks comprised of multiple depthwise-separable convolutions, followed by multi-head self-attention and feed-forward layers. Due to the size and complexity of the architecture, we focused on a modular implementation of QANet, so that the configuration could be parameterized and the model could be scaled in terms of number of convolutional layers, encoder blocks, self-attention heads, etc. The model size was reduced (as per [2]) via weight-sharing within multiple instances of stacked model encoder blocks and stacked embedding encoder blocks for both the question and context.

![Figure 1: Depthwise convolutions with Multiple Convolutional Kernels (3, 5, 7)](image)

We enhanced the QANet [2] model’s capacity by using multiple convolutional kernels in the encoder block. We used three kernel sizes (3, 5 and 7), and used a convolution projection layer to combine those outputs into a latent embedding to be passed into the subsequent layers in the encoder block. The motivation for this design choice is to make the model capable of learning multiple n-gram (3, 5 and 7) phrasal features in the embedding encoder block. This design choice enables various combinations of latent features in the model encoder block.

### 3.3 Answer Verification: Abstention Decoder Module

A key challenge of the extractive reading comprehension task for the SQuAD 2.0 dataset is knowing when to abstain from answering a question. We experimented with two approaches aimed at improving our models’ ability to classify a (context, query) pair based on whether an answer to the query is present in the context. Both of the methods presented below can be applied to any base model, and we experimented with applying them both to our BiDAF and QANet models.

#### 3.3.1 Abstention Head

We first experimented with augmenting the baseline BiDAF model with an “abstention head”, in addition to the two heads predicting the start and end of the answer span, and added an extra term to the loss function to penalize incorrect answer versus no-answer predictions (using binary cross-entropy loss). Using this approach, the answer/no-answer probability was calculated jointly with the span prediction as in the baseline, and then again independently using the abstention head with no shared parameters:

$$\mathcal{L} = \mathcal{L}^{\text{joint}} + \lambda \mathcal{L}^{\text{ans}},$$

where $\mathcal{L}^{\text{joint}}$ is the negative log-likelihood (NLL) loss incurred by the baseline BiDAF model, $\mathcal{L}^{\text{ans}}$ is the binary cross-entropy loss incurred by the abstention head, and $\lambda$ is a hyperparameter for which
we tried values in the set \{0.5, 1, 1.5\}. Note that for a ground-truth no-answer example, the total loss is given by \( NLL(s_0) + NLL(e_0) + \lambda L_{\text{ans}} \), where \( s_0 \) and \( e_0 \) are the softmax scores of the span start and end predictions at index 0 (no-answer location), respectively, jointly normalized with the span scores across all positions in the given context. This approach was inspired by [14], which showed evidence of over-confident softmax probabilities on answer span predictions when jointly normalized with no-answer predictions (as in baseline BiDAF). We will refer to this as the confliction problem. Since the sum of the normalized span and answer/no-answer scores is fixed at 1, inaccurate confidence on answer span probabilities could lead to imprecise prediction of no-answer scores. The relatively low baseline AvNA score of about 68%, gave further evidence that the model was learning how to predict spans better than it was learning when to abstain.

The abstention head was designed as follows:

1. We passed outputs \( \text{att} \) and \( \text{mod} \) from the context-query attention and model encoder layers, respectively, through linear layers with separate parameters, then added the results, yielding an output tensor of size (batch_size, sequence_length, hidden_size).
2. We then performed masked pooling over the sequence_length dimension of this tensor. We experimented with mean pooling, max pooling, and mean-then-max pooling. The output of the pooling layer had dimension (batch_size, hidden_size).
3. We applied a final linear layer with an output dimension of 1, yielding a tensor of size (batch_size, 1), then took the sigmoid of each element. The sigmoid of the abstention logits was then passed to the loss function along with the span softmax scores.

3.3.2 Verifier Model

The decision of when to abstain depends upon a logical entailment relationship between the query and proposed answer. The SQuAD 2.0 dataset contains many examples of questions for which a plausible answer is present in the context, but this answer either contradicts the question or fails to address the question is asking. This subtle relationship cannot be gleaned from simple dot product-based similarity or attention-based metrics between the query and answer, since the answer can highly resemble the query, but still contradict it. Thus, we experimented with building a two-stage pipeline, starting with an initial “Reader” model, which consisted of an augmented version of our base BiDAF/QANet models, and a “Verifier”, which was trained separately from our Reader, and whose job it was to detect entailment relationships given a query and plausible answer, yielding the probability that the plausible answer is a true answer to the question. Our approach of using separate modules for reading and answer abstention verification, as well as the loss function used in our Reader, was inspired by [5], though the Reader and Verifier architectures we use differ from those presented in the paper. In particular, [5] uses the Reinforced Mnemonic Reader [15] as the Reader and uses LSTM-based embedding encoder and model encoder blocks in the Verifier, whereas our Verifier is fully non-recurrent, and uses convolutions + self-attention in the encoder blocks. To train both our Reader and Verifier, we wrote a custom data loader which included plausible answers for no-answer questions, as described in Section 4.1.

**Reader:** We converted our baseline BiDAF and QANet models into Readers for our Read + Verify pipeline by adding in two additional output heads and modifying our loss function. Rather than adding an independent abstention head, as in Section 3.1.1, we added two independent “span heads” to predict the start and end indices of an answer, respectively, with no option for predicting no-answer. These heads had the same architecture as the original BiDAF and QANet heads, but did not share parameters, and were designed to aid in plausible answer extraction, while largely delegating the task of determining whether the plausible answer was a true answer to the Verifier.

As in Section 3.3.1, we added two independent terms to the loss function, in addition to the joint loss, in order to address the confliction problem (described above) which is inherent to multi-task settings. Let \( S \) and \( E \) represent the baseline BiDAF/QANet model’s start and end heads (joint normalization with the no-answer prediction), and let \( \tilde{S} \) and \( \tilde{E} \) represent the new start and end heads we introduced, without a no-answer prediction. Let \( \alpha \) and \( \beta \) be the span scores for the start and end indices in \( S \) and \( E \), respectively, and let \( \tilde{\alpha} \) and \( \tilde{\beta} \) represent the span scores for the start and end indices in \( \tilde{S} \) and \( \tilde{E} \), respectively. Then let \( z_s = \alpha[0] \) and \( z_e = \beta[0] \) denote the (unnormalized) no-answer score given by the logits at index 0 in \( S \) and \( E \)’s start and end answer span predictions,
respectively. Let $a$ and $b$ be the ground-truth start and end indices, if a query is answerable, and let $\hat{a}$ and $\hat{b}$ represent the start and end indices of the ground-truth plausible answer. Finally, let $l_c$ be the length of the context, let $\delta$ equal 1 if a question is answerable and 0 otherwise, and let $\sigma$ be the sigmoid function. Then we construct our loss function as the sum of joint and marginal effects, given our objective of predicting span start and end indices if there is an answer, and no-answer otherwise:

$$L = L_{\text{joint}} + \gamma L_{\text{span}} + \lambda L_{\text{ans}},$$

where

$$L_{\text{joint}} = -\log \left( \frac{(1 - \delta) e^{z_s z_e} + \delta e^{\alpha_a \beta_b}}{e^{z_s z_e} + \sum_{i=1}^{l_c} \sum_{j=1}^{l_c} e^{\alpha_i \beta_j}} \right),$$

$$L_{\text{span}} = -\log \left( \frac{e^{\hat{\alpha}_{\hat{a}} \hat{\beta}_{\hat{b}}}}{\sum_{i=1}^{l_c} \sum_{j=1}^{l_c} e^{\alpha_i \beta_j}} \right),$$

and

$$L_{\text{ans}} = -(1 - \delta) \log \sigma(z) - \delta \log(1 - \sigma(z)).$$

Note that $L_{\text{joint}}$ is the original BiDAF objective, which is the negative log-likelihood of a shared softmax function applied to normalize both the no-answer score and span scores. The independent no-answer binary cross-entropy loss $L_{\text{ans}}$ is computed from the sigmoids of the (unnormalized) logits provided by the $S$ and $H$ heads (unlike in Section 3.3.1 where the independent no-answer loss has separate parameters). This loss term helps offset any weakening in no-answer detection due to the joint normalization, but does not perform the heavy lifting reserved for the abstention Verifier. Finally, note that the ground-truth labels used to calculate $L_{\text{joint}}$ are distinct from those used to calculate $L_{\text{span}}$ for all unanswerable queries, since in the former case, the ground-truth start and end indices $a$ and $b$ are 0, while in the latter case, $\hat{a}$ and $\hat{b}$ are the starts and ends of the plausible answers.

**Verifier:** Our Verifier model was composed of the following basic building blocks: embedding layer, encoder, inference modeling inter-attention block, model encoder with intra-sentence modeling, concatenation of proposed answer and query representations, and a final feedforward + sigmoid layer. The detailed architecture is depicted in appendix.

The final prediction of our Read_Verify model was computed as follows: If the average of the no-answer probability predicted by our Reader and Verifier models exceeded threshold (a hyperparameter tuned on the validation set), then our final prediction was also no-answer. Otherwise, we predicted the answer span given by heads $S$ and $E$ of our Reader model.

### 4 Experiments

#### 4.1 Data

We used the SQuAD 2.0 dataset [16], which differs from SQuAD 1.0 in that about $\frac{1}{3}$ of questions contain no valid answer in the provided context. For our baseline models, we trained and tested using the data loader provided in the default project starter code. We then wrote a custom setup.py file to extract plausible answers for the no-answer questions from the raw dataset, while keeping track of which questions were unanswerable. Note that in the raw SQuAD 2.0 dataset, plausible answer(s) have been annotated for every unanswerable question, but this data is ignored by the default setup.py script. The dataset has 129,941 training examples, 5915 examples in the development set, and 5915 examples in the test set.

We used our custom data loader to train the Reader and Verifier models, since the downstream Verifier, whose job is to confirm or deny textual entailment, depends upon being provided with candidate answers for unanswerable questions. Thus, the context and query in both datasets that we used were identical; it was only the set of (plausible) answers that changed. These answers were used in the loss function for our Reader models, and were used as input to train our Verifier model.

#### 4.2 Evaluation method

We used Exact match (EM), F1 score and AvNA as metrics for our experiments. Exact Match is a binary metric (i.e. true/false) computing the correctness of predicted answer span such that it matched the ground truth answer exactly. An exact match gets an EM score of 1 and a mismatch gets an EM score of 0, for a given example. F1 score is a less strict metric; it is the harmonic mean
of precision and recall. AvNA (Answer vs No-Answer) score measures the correctness of binary Answer/No-Answer predictions of the model, with respect to the ground truth. Although the baseline QANet model does compare the speed-up against the RNN-based models by replacing the QANet’s CNN layers with RNN layers, we chose not to focus on this metric, in the interest of exploring enhancements over the baseline model. The speed-ups would be gained by virtue of CNN’s and layer-dropouts [17] used in the QANet architecture.

4.3 BiDAF-based models

When training all BiDAF-based models, we used a learning rate of 0.5, an exponential moving average decay rate of 0.9999, L2 regularization with $\lambda = 3 \cdot 10^{-7}$, and a hidden size of 100. The remaining hyperparameters we explored are detailed in Figure 2.

4.4 QANet-based models

4.4.1 Training details

For our QA training we used Tesla K80 GPUs, on Azure NC6 and NC12 Virtual machines. Due to the limited size of GPU VRAMs (11 GB), we had to adjust our batch size, attention head count and hidden-size to fit the model parameters in the GPU memory. We tried both Adam and AdaDelta optimizers. Based on the initial performance of the model, we used Adam for QANet-based model training. For the Adam optimizer, we used $\beta_1 = 0.8$, $\beta_2 = 0.9999$, $\epsilon = 10^{-8}$ and weight decay $= 3 \cdot 10^{-7}$. We used a learning rate warm-up scheme with inverse exponential increase from 0.0 to 0.001 in 1000 training steps, and kept the learning rate constant afterwards. We also used layer-dropout based stochastic depth method within encoder layers with the dropout probability for layer $l$ computed as $p_l = 1 - l \cdot (1 - p_L)/L$, where $L$ is the last convolution layer (count) and $p_L = 0.9$.

4.4.2 Hyperparameter Search

We did hyperparameter search across learning-rates (0.0007 to 0.005), hidden size (96, 128), character embedding size (96, 128), and number of self-attention heads (1, 4, 8).

4.4.3 Ablation Studies

For the QANet [2] model, we performed ablation studies in two areas: multi-head attention and learning-rate warm-up scheme.

- Multi-head attention is a key component of QANet encoder blocks and we experimented with attention head count. The F1 and EM graphs in Figure 4 show that the model with 4-attention heads converges faster than the single attention head model. In this experiment the batch-size for the 4-head attention model and one attention head model were 16 and 32 respectively, in order to fit the model within GPU memory.

- We observed that with learning warm-up, the initial steps have slow progression but eventually the learning converges and it doesn’t seem to have a significant impact on the final performance of the model. This observation could be attributed to the non-skewed nature of the training data produced by the data loader, which is purely incidental.

4.5 Reader-Verifier Models

The hyperparameters used to train our Reader models were identical to those described in Sections 4.5 and 4.6. We trained our Reader and Verifier models separately, then used them to predict jointly at test time. At both train and test time, the input to our Reader was the (context, query) tuple, and its output was the predicted answer span from heads $S$ and $E$, as well as from heads $\tilde{S}$ and $\tilde{E}$. In contrast, the train time and test time routines for the Verifier differed. We adopted a “teaching forcing” routine at train time, in which the Verifier was fed the ground-truth plausible answer sentence, along with the query, and predicted the probability that the plausible answer was indeed a true answer to the query. At test time, the Verifier was instead fed the answer sentence corresponding to the answer span predicted by the Reader. The final threshold we used for no-answer predictions was 0.5, 0.85 for BiDAF_Reader and QANet_Reader, respectively.
5 Results

Our best model based on QANet scored 66.07 F1 and 62.434 EM scores on the test leaderboard. Our detailed dev set results for both BiDAF- and QANet-based models are presented in the subsequent sections.

5.1 BiDAF

Out of all the BiDAF-based models, the BiDAF_Reader performed the best, while the BiDAF models with abstention heads performed the worst. This suggests that independently predicting answer spans and the answer/no-answer probability performs worse than joint predictions, but that including additional, independent span prediction heads can actually enhance learning. Thus, reading comprehension with abstention is well-suited to a multi-task framework, in which parameters for the two related tasks are learned together. That being said, augmenting with independent span heads in addition to the multi-task heads can lead to improved performance.

5.2 QANet

Our baseline QANet model has 1 attention head, 96 hidden size, 96 character embedding size, 300 word embedding size, learning rate of 0.001 and 32 batch size during training. We used dropout rate of 0.1 across all the layers except the embedding layer, where dropout rate was 0.2. We noticed the that the F1 score of QANet model with 4 attention heads, is same with our baseline model. However, the NLL score and EM scores are lower with 4 attention heads. This could be explained by the smaller batch size of 16 that we had to use for the 4-attention heads based model training as opposed to batch size of 32, for 1-attention head training. So, it can be argued that the 4-attention head model could perform better, if attempted with more number of epochs with slightly lower learning rate than 0.001. Our results indicate that our performance metrics improved with the multiple-kernel feature, namely F1 score improved from 68.03 to 68.16.

Our baseline Read_Verify model is with 1 attention head, and the embedding parameters for Verifier are same as of the baseline QANet model. The learning scheme used is learning warm-up with 1000 warm-up steps and learning rate of 0.001. The char-embedding dimension used was 96, and model hidden dimension was 96. The Verifier model was integrated with both BiDAF_Reader and QANet_Reader. The performance of the Read_Verify model did not exceed the best performance of the QANet model. We did see that the model was very sensitive to answer/no-answer threshold and after even after fine-tuning we only got best score as 65.32 F1, 71.16 AvNA and 62.39 EM.

<table>
<thead>
<tr>
<th>BiDAF-based results</th>
<th>NLL loss</th>
<th>F1</th>
<th>EM</th>
<th>AvNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiDAF</td>
<td>3.15</td>
<td>61.01</td>
<td>57.54</td>
<td>68.19</td>
</tr>
<tr>
<td>BiDAF + char_embeddings</td>
<td>2.91</td>
<td>63.82</td>
<td>60.19</td>
<td>70.66</td>
</tr>
<tr>
<td>BiDAF + char_embeddings + abstention head (δ = 1)</td>
<td>3.93</td>
<td>62.75</td>
<td>59.33</td>
<td>69.70</td>
</tr>
<tr>
<td>BiDAF + char_embeddings + abstention head (δ = 0.5)</td>
<td>5.69</td>
<td>62.05</td>
<td>58.68</td>
<td>68.76</td>
</tr>
<tr>
<td>BiDAF + char_embeddings + abstention head (δ = 1.5)</td>
<td>3.76</td>
<td>62.00</td>
<td>58.46</td>
<td>69.25</td>
</tr>
<tr>
<td><strong>BiDAF_Reader</strong></td>
<td><strong>4.35</strong></td>
<td><strong>65.29</strong></td>
<td><strong>62.06</strong></td>
<td><strong>71.46</strong></td>
</tr>
<tr>
<td>BiDAF_Reader + QANet_Verifier</td>
<td>5.63</td>
<td>63.64</td>
<td>60.61</td>
<td>69.95</td>
</tr>
</tbody>
</table>

(a) Comparison of development set results across NLL, EM, F1, and AvNA evaluation metrics for the BiDAF-based models

<table>
<thead>
<tr>
<th>QANet-based results</th>
<th>NLL loss</th>
<th>F1</th>
<th>EM</th>
<th>AvNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>QANet, heads=1</td>
<td>2.56</td>
<td>68.03</td>
<td>64.46</td>
<td>74.00</td>
</tr>
<tr>
<td>QANet, heads=4</td>
<td>2.77</td>
<td>68.03</td>
<td>64.39</td>
<td>74.24</td>
</tr>
<tr>
<td>QANet, heads=8 (10 epochs)</td>
<td>2.55</td>
<td>65.74</td>
<td>62.51</td>
<td>71.35</td>
</tr>
<tr>
<td><strong>QANet, heads=1, multiple kernel sizes (3, 5, 7)</strong></td>
<td><strong>2.579</strong></td>
<td><strong>68.16</strong></td>
<td><strong>64.75</strong></td>
<td><strong>74.51</strong></td>
</tr>
<tr>
<td>QANet_Reader</td>
<td>4.34</td>
<td>66.83</td>
<td>63.40</td>
<td>73.30</td>
</tr>
<tr>
<td>QANet_Reader + QANet_Verifier</td>
<td>5.56</td>
<td>65.32</td>
<td>62.39</td>
<td>71.16</td>
</tr>
</tbody>
</table>

(b) Comparison of development set results across NLL, EM, F1, and AvNA evaluation metrics for the QANet-based models

Figure 2: BiDAF and QANet based model performance metrics on SQUAD2.0
6 Analysis

In the initial iterations of training, the model seems to generate No-Answer predictions (Figure 6a). Around the completion of 8-10 epochs, the model starts predicting meaningful spans but is still not resolving co-references correctly. Interestingly, few predictions contain an incorrect answer span when there is no answer, as shown in Figure 6b. There are certain cases where the model predicts the answer span close to the ground truth, although the answer span doesn’t contain the explanatory detail that would have made the answer more complete. At the same time the ground truth of the devset in such cases, lacked the explanatory details in their answer span as well, as shown in Figure 6c. There are a few answers that are incorrect, and close investigation shows that the attention is deficient in determining hierarchical relationships between constructs. For example in this Figure 6d case, the geographical construct ‘dunes’ are hierarchically part of ‘Vistula terraces’.

Over the course of this project, we learned that the self-attention mechanism is highly effective in capturing latent features, and that the self-attention improves with number of attention heads. Secondly, we observed that performance was correlated with the number of parameters and varied projections into representative sub-spaces. Thirdly, from Read_Verify experiments’ performance below expectations, we observed that answering some questions requires reading the context span larger than the plausible answer context; sometimes the full context is needed. Lastly, in terms of practical constraints for training, we realised that there are trade-offs between batch size and attention head count, in order to fit the model in GPU memory.

7 Conclusion

This project helped the team gain critical insights into abstention-aware reading comprehension architectures. The QANet results demonstrate that, despite their non-sequential structure, CNN-based architectures are capable of effectively performing reading comprehension tasks. Our contributions include the addition of abstention heads to our baseline models, and of multiple kernel sizes to QANet. The BiDAF_Reader achieved a significant improvement of 1.4 on F1 score, over other BiDAF-based models in our experiment. Moreover, the multiple-kernel enhancement over QANet improved the performance on F1, EM and AvNA scores. As future work, we would propose to extend the application of beam search to augment the inter-attention layer, such that multiple inference outputs could be carried over to the model encoder layers. Our hypothesis is that similar to beam search this approach would allow for additional candidates to be evaluated for prediction, thus generalizing the model’s capability further.

References


A Appendix

A.1 BiDAF and QANet models training

![Figure 3: Results [QANet + BiDAF models] (BiDAF-reader, QANet Multiple-kernels, QANet-dim128, QANet-dim128, QANet-LR-0.003)](image-url)
A.2 BiDAF based model predictions

(a) Incorrect No Answer Prediction-early training

(b) Correct prediction-dissimilar answer span

(c) Incorrect Answer Prediction-early training

(d) Incorrect answer prediction span

Figure 5: BiDAF model prediction on development data-set sample over the course of training

A.3 QANet based model predictions

(a) Incorrect No Answer Prediction-early training

(b) Incorrect Answer Prediction-early training

(c) Correct prediction-dissimilar answer span

(d) Incorrect answer prediction span

Figure 6: QANet model prediction on development data-set sample over the course of training
A.4 Verifier Architecture

![Verifier Architecture](image1.png)

Figure 7: Verifier Architecture motivated by QANet

A.5 Reader-Verifier Test Architecture

![Reader-Verifier Test Architecture](image2.png)

Figure 8: Reader-Verifier Architecture in Test Mode