QANANET: Improve Question Answering By Learning Not To Answer

Stanford CS224N Default Project

Zixiao Ken Wang
Stanford Center for Professional Development
Stanford University
zixiaow@stanford.edu

Abstract

Accurate automated question answering help people learn new knowledge at large scale. In this project, we attempt to solve the question answering task proposed by the SQuAD dataset by 1) increasing accuracy of Bi-directional Attention Flow (BiDAF) [1] using convolution and self-attention based on QANet [2] and 2) adding a new classification AvNA head and loss function for QANet to explicitly handle non-answer case which are new to SQuAD 2.0 [3]. This report shows that our QANet implemented from scratch significantly improves the baseline BiDAF model F1 dev score from 60.71 to 69.34, and ExactMatch (EM) from 57.10 to 65.52. Our final model with AvNA head further improves the dev score to 70.37 for F1 and 66.85 for EM. On the IID default track test leaderboard, our model achieves relatively high F1 of 66.581 and EM of 62.975.

1 Key Information to include

• Mentor: Kaili Huang
• External Collaborators (if you have any): No
• Sharing project: No

2 Introduction

Question and answering, or machine reading comprehension in general, is relevant to every modern person’s life. We search different questions on Google everyday for answers related to areas such as historical facts, a particular academic subject, or recent news. Recent advancement in deep learning and natural language processing has allowed question and answering systems such as Google to serve questions more accurately at a larger scale. However, no questioning and answering system is perfect. Figure 1 illustrates an answer provided by Google for one of the questions included in the SQuAD 2.0 dataset. In this case, Google did not answer the question correctly by suggesting “rice” coming to western Japan but the user is more likely looking for a crop with western country origin.

In addition to providing hard questions, SQuAD measures how well our system is able to extract the correct span of texts based on different contexts for the same question. To increase the difficulty of the problem, the SQuAD 2.0 authors created 50k unanswerable questions out of 150k total questions by adversarially injecting confusing context based on crowd-sourced data. For example, the same crop question will be given the context of crop coming from western country to China instead of Japan, and the system should label the question and answer pair as no-answer. Therefore, a good question and answering system for SQuAD 2.0 should learn not to answer certain questions when the given context is not relevant.

In this report, we demonstrate that our system QANAEAT improves the baseline BiDAF model qualitatively by determining that questions similar to the crop example cannot be answered given the
wrong context. We will also show how we improve the system quantitatively to achieve relatively high \( F1 \) score of 66.581 and \( EM \) of 62.975 on the SQuAD test leaderboard.

![Image of Google answering user questions.](image)

Figure 1: An example of Google answering user questions. The same question is included as part of the SQuAD 2.0 dataset.

3 Related Work

Before the introduction of Transformer \(^4\), most state-of-the-art automated question answering system for SQuAD used recurrent model such as LSTM to process question. Our baseline model Bi-directional Attention Flow (BiDAF) \(^1\) introduced attention between question and context and achieved state-of-the-art score for the original SQuAD dataset which did not contain any unanswerable question. Other similar end-to-end neural networks model for reading comprehension style question answering include R-Net \(^5\) and DCN \(^6\).

The RNN based approaches has issues with referring to previous words or contexts in long sequences while attending to local interaction of each word. Our project improves the baseline by replacing RNN with convolution and self-attention based on the network design of QANet \(^2\). QANet was the state-of-the-art model for SQuAD before the introduction of BERT \(^7\). We chose QANet as our foundation for improving SQuAD 2.0 performance instead of BERT because BERT requires large corpus of texts for pretraining followed by finetuning with SQuAD training set while QANet can be trained directly using the SQuAD training set. Using large corpus, or any other dataset, for pretraining does not fulfill our project requirement and therefore, QANet is the best choice.

Besides building QANet to improve the overall SQuAD performance, we predicts the no-answer probability for unanswerable questions in SQuAD explicitly using an AvNA (answer versus no-answer) head. As far as we know, this is the first of its kind solution attempting to improve the no-answer prediction.

4 Approach

4.1 Use QANet to improve baseline

Our main approach builds QANet \(^2\) from scratch using the QANet paper as the source. QANet shares the same general structure as our baseline BiDAF \(^1\). To replace the RNN function, QANet uses embedding encoder and model encoder with convolution and self-attention. The self-attention design follows the design of transformer \(^4\) and uses positional embedding added to the original input to handle sequencial information. QANet uses similar word and character embedding setup, context-query attention, and output layer as BiDAF (please refer to the Appendix section for detailed comparison between QANet and BiDAF).

4.2 AvNA HEAD

Besides implementing QANet, as an original contribution, we explicitly handle the no-answer case of SQuAD 2.0 by adding a classification AvNA head output for predicting the probability of no-answer given question and context. The experiment section will discuss different network designs for the
AvNA head and our final model design is illustrated by Figure 2, where the block with AvNA head marks our new addition to the existing QANet architecture shown by the rest of the figure. The QANet input and output and its architecture is not changed to support the extension ((please refer to the Appendix section for detailed explanations for each QANet layer).

Our AvNA head uses 3 Conv1D layers with kernel size 1 and decreasing number of output channel sizes of 128, 64, and 1. The input channel for the first layer has size of 384 by concatenating all outputs $M_1$, $M_2$, and $M_3$ from the QANet Model Encoder layer with each output operating with hidden size of 128. If our text sequence has size $T$, which makes our input shape for AvNA head $(T, 384)$, our last Conv1D layer output has the shape of $(T, 1)$. The probability prediction for no-answer only requires 1 probability number. Therefore, we follow the common practice of taking the last hidden element of the sequence $(T, 1)$ as our output with shape $(1, 1)$. The output of Conv1D layers is then passed to a Sigmoid function to produce value with the range of 0 to 1 as our final prediction $p_3$ for the probability of the question with no answer based on the context.

For the new learning object, we use the binary cross entropy loss between no-answer prediction $p_3_i$ and its golden label $y_3_i$ for example $i$:

$$L(\theta)_{NA} = -\frac{1}{N} \sum_i binary\_cross\_entropy(p_3_i, y_3_i)$$  \hspace{1cm} (1)$$

where $y_3_i$ is the true label of 1 for the $i$th example with no-answer, otherwise 1, and $p_3_i$ is the prediction for the new classification head. $\theta$ is the model parameter, and $N$ is the number of examples.

This loss function can be trained separately as the only loss function for our model to produce $p_3$. Alternatively, we can train the loss function jointly with the original QANet loss function which measures the cross entropy loss between the predicted start position $p_1_i$ and gold start label $y_1_i$, and the predicted end position $p_2_i$ against gold end label $y_2_i$ for each example $i$:

$$L(\theta)_{QANet} = -\frac{1}{N} \sum_i [cross\_entropy(p_1_i, y_1_i) + cross\_entropy(p_2_i, y_2_i)]$$  \hspace{1cm} (2)$$

$$L(\theta)_{JOINT} = L(\theta)_{QANet} + \lambda L(\theta)_{NA}$$  \hspace{1cm} (3)$$
where $\lambda$ is a tuneable hyperparameter to control the balance between predicting no-answer and predicting answer in with the right start and end position.

### 4.3 Baselines

We use BiDAF [1] as default baseline without any modifications to the default project starter code.

### 4.4 Code Reference

We implemented QANet on top of BiDAF with minimum reference to the publicly available QANet Pytorch code. We took the commonly available depthwise separable convolution function to replace our nn.Conv2d and their implementation of sine and cosine positional encoding which is similar to the one used by transformer [4]. We also took inspirations for layer dropout and character embedding but the actual integration is our own.

For self-attention, we used the CausalSelfAttention from our assignment 5 which has one fixed length mask. We modified it to support different masks for question and context sequences with different max lengths. We also experimented with the self attention module from QANet Pytorch code and modified the implementation of mask to account for our default padding setup.

The rest of the code including but not limited to all changes to the structure of embedding encoder, model encoder, output layers are implemented using QANet paper as the sole reference. The implementation for the AvNA classification head, joint training, model ensemble and the new classification loss is our original work.

### 5 Experiments

#### 5.1 Data

We use the CS224N default project SQuAD train/dev dataset for training and fine-tuning. We adapted all BiDAF input and output without modification to support QANet. To create answer or no-answer label $y_3^i$ for the $i$th example, we simply check if the corresponding start position label $y_1^i$ and end position $y_2^i$ is 0, which is the position of the OOV (Out of Vocabulary), because after the baseline BiDAF data preprocessing, any example with start and end position label 0 means no-answer.

#### 5.2 Evaluation method

We use the default project $EM$ (Exact Match), $F_1$ and $AvNA$ (Answer versus no Answer) scores as evaluation metrics. At the high level, $EM$ compares the exact text for predicted answer and answer label, $F_1$ is the harmonic mean of precision and recall, and $AvNA$ evaluates the classification accuracy of the model when predicting answer or answer. Please review [8] for details.

#### 5.3 Experimental details

##### 5.3.1 QANet

Following the QANet architecture, our the encoding layer uses 1 encoding block with 4 conv layers, kernel size 7, and 128 filters. For the model encoding layer, we use 7 encoding blocks each with 2 conv layers, kernel size 5 and 128 filters. For self-attention, we use 8 multi-head attention heads for both embedding encoder layer and model encoder layer. And for the context-query attention, we started with the BiDAF attention but later switched to the publicly available QANet Pytorch’s attention implementation. The main difference is that the latter uses Conv1D with kernel size 1 instead of linear layer as the feedforward layer. For consistency, we use Conv1D with kernel size 1 for all components that require linear layer. Throughout the model, we modified the original hidden size from 100 to 128.

For training, we changed the optimizer from Adadelta to Adam with $\beta_1 = 0.8$, $\beta_2 = 0.999$, $\epsilon = 1e^{-8}$, and weight decay $3e^{-7}$, and decreased the learning rate from 0.5 to 0.001. We use learning rate warm up with inverse exponential increase from 0.0 to 0.001 and keep constant learning
rate after the first 1000 steps. Our dropout rate is 0.1. We referenced both the publicly available QANet Pytorch code and QANet paper when finding the best parameters.

We initially trained the QANet with character embedding of 64 dimension with 30 epochs but later switched to 50 epochs for the QANet with character embedding of 200 with the best checkpoint selected at epoch 43 based on F1 score. All data in this report is trained with batch size 36 with Ubuntu 18.04 and PyTorch 1.10 using a single Nvidia RTX 3090 with training speed around 150 iterations per second. We also attempted to use Microsoft Azure NC6s_v3 instance for training but the results are only intermediary and not reported here.

5.3.2 AvNA Head

Our final AvNA head training uses learning rate 0.00001 for finetuning based on best weights trained for QANet. The finetuning process uses only the binary classification objective \( L(\theta)_{NA} \) (equation 1) as loss function, and converges after 10 epochs and we selected the best weights based on AvNA because it is only intended for improving answer or no-answer prediction. Producing the AvNA binary prediction requires setting a threshold for the no-answer probability \( p_3 \). We used threshold of 0.2 during training and model selection. Although we only finetuned \( p_3 \), our finetuned weights can produce reasonable prediction for \( p_1 \) start position and \( p_2 \) end position. We report the whole model with \( p_1 \), \( p_2 \) and \( p_3 \) outputs as QANANET single model.

5.3.3 Ensemble Model

To ensemble the model for final submission, we load the finetune weights from the QANANET single model and use them to predict no-answer probability \( p_3 \), and use another model with best QANet weights to predict \( p_1 \) and \( p_2 \) only if there is answer to the question determined by \( p_3 \). To find the best \( p_3 \) threshold, we run inference with different thresholds through grid search, and select the final model for testing based on the dev score. The final threshold for \( p_3 \) is 0.14. We report the results for this model as QANANET ensemble model.

5.4 Results

5.4.1 Official Scores

<table>
<thead>
<tr>
<th>Leaderboard</th>
<th>F1</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev</td>
<td>70.365</td>
<td>66.846</td>
</tr>
<tr>
<td>Test</td>
<td>66.581</td>
<td>62.975</td>
</tr>
</tbody>
</table>

Table 1: Accepted official dev and test IID track default project score for our QANAET ensemble model.

Our final QANAET ensemble model scores F1 of 66.581 and EM of 62.975 on the official test leaderboard for the IID track default project. Table reports our accepted scores for the dev and test leaderboard. Our model achieves reasonably high rank among the participants.

One common issue among all participating models is that the test scores are always a few points below corresponding dev scores. Because we do not have access to the test dataset labels, we could only hypothesize that the data distribution for the test dataset is slightly different from train or dev dataset to make the problem harder. The reduction in score could also mean that our model tune on dev data is not able to generalize to all test data. In addition, because our final QANAET ensemble model has an extra threshold term for tuning, we might be overfitting to the dev set during the final grid search.

5.4.2 Experiment 1: Model Comparison Based On Dev Scores

We compared the some of our best performing models against our BiDAF baseline and summarized the results in Figure. Our implementation of QANet can significantly increase the baseline F1 score from 60.71 to 69.32, EM score from 57.1 to 65.52, and AvNA from 67.84 to 75.9.

1 top 20% based on F1 or EM test score as of 03-14-2022
Adding AvNA head further boosts the QANet score to 70.05 for F1, 66.43 for EM and 76.49 for AvNA. The increase in score is more moderate than we expected. We suspect that QANet is already capturing the no-answer concept very well using the OOV token as prediction for the $p_1$ starting and $p_2$ ending position. Nevertheless, the increase in scores proves that our design of AvNA head and binary classification learning objective is beneficial. In addition, the single QANANET model achieves the highest AvNA score. This is not surprising because increasing AvNA was the sole objective of AvNA head finetuning.

Finally, our ensemble model takes the best of both models by combining our best QANet and single QANANET for inference, and achieves the best dev F1 score of 70.37, and EM score of 66.85, and slightly lower AvNA of 76.26 comparing to the best score of 76.49 from the single QANANET model. The decrease in AvNA score is due to the fact that we are using a more restrictive threshold of 0.14 when predicting answer or no-answer for the ensemble model while the single model uses 0.2 as threshold. Using 0.14 threshold for the single QANANET model will yield the same AvNA score as the ensemble model and a small increase in F1 from 70.05 to 70.20, and EM from 66.43 to 66.80 which are smaller comparing to the corresponding scores from the ensemble model.

![Figure 3: Dev score comparison based on a few of our best models against baseline](image)

**5.4.3 Experiment 2: Ensemble Model Threshold Selection**

Choosing the correct ensemble model requires binary prediction threshold selection for the AvNA no-answer head $p_3$. By using a smaller threshold, we allow more answers in our final submissions which may lead to more questions without answer classified as with answer. A larger threshold number can lead to less question without answer as our prediction but the prediction for no-answer could be more accurate.

We use grid search from 0.05 to 0.9 to find the correct threshold for our dev dataset. Because grid search does not require training, we can run it quickly by inferencing the same model multiple times with varying threshold when predicting answer no-answer. Figure 4 shows the trend of score changes as we increase the thresholds. We determined that 0.14 is the best threshold value based on its good balance among F1, EM and AvNA scores. The value of 0.09 is the threshold value that leads to the highest F1 of 70.42, EM of 67.15 and slightly lower AvNA 75.70. We did not get the chance to test this value for test leaderboard because it is limited to 3 tests per team.

One last consideration related to threshold selection is that we could slightly modify the original QANet no-answer prediction to achieve automatic threshold selection. If the probability $p_3$ is greater than any predicted answer span determined by start position probability $p_1$ and end probability $p_2$, the model predicts no-answer (see [8] for more details on how QANet make no-answer prediction). Using this strategy, our model achieved lower EM of 65.367(−1.479) and F1 of 69.233(−1.132) for the dev set, and EM of 62.502(−0.473) and F1 of 66.181(−0.400) for the test set comparing to ensemble model with threshold 0.14.
Figure 4: Ensemble model no-answer threshold grid search. The choice of threshold values affects $F_1$, $EM$ and $AvNA$ scores. All scores are reported based on dev dataset evaluation. For visualization purpose, the x-axis is not evenly divided for the threshold values.

5.4.4 Experiment 3: QANet Ablation Study

Our implementation of QANet went through multiple iterations and for almost every iteration, we added new components that help improve the final scores. Figure 5 summaries the score changes based on our major iterations.

Specifically, most of our score improvements come from character embedding. We found that adding frozen character embedding can add significant $2.43$ to $F_1$ score with similar improvements for $EM$ comparing to QANet baseline. Making character embedding trainable after adding dropout and layer dropout regularization help increase $F_1$ and $EM$ scores significantly. And increasing the character embedding size from the BiDAF default of 64 dimension to 200 produces our best model for QANet.

Regularization and learning rates are also important. Changing the optimizer to ADAM with learning rate tuning, we achieve slightly better $F_1$ score and similar improvement for $EM$. However, the $AvNA$ score is lower. Adding regularization such as dropout and layer dropout detailed in [2] help further increase $F_1$, $EM$ and $AvNA$ scores. Note that for consistency, we modifies all linear layers to $Conv1D$ with kernel size of 1 as part of the “+dropout and conv1d” regularization change. We do not expect the change has major impact but it could lead to different results because the weights for $Conv1D$ layers are initialized differently than the Linear layers. The change also include QANet Pytorch implementation of self-attention and the only major difference is that our previous self-attention based on mini-gpt uses Linear layers while the new self-attention uses $Conv1D$ layers.

Trainable positional encoding based on mini-gpt did not help increase our score. Our best model uses the same sine and cosine fixed positional encoding as Transformer [4]. Regardless of the dev set results, using trainable positional encoding in this project will not generalize dev results to test dataset because our test data has longer maximum paragraph and context and answer length. In other words, during inference for test, our model with trainable positional encoding might give random encoding to the part of texts beyond the maximum length of dev dataset texts, which would lead to unpredictable test scores.

5.4.5 Experiment 4: Best AvNA head design

The following study compares alternative designs and training methods for the AvNA head. This study is conducted using character embedding dimension of 64 before we switch to character embedding of 200. Therefore, the scores here are not comparable to our final model scores.

**AvNA Head Small (no OOV):** We can train AvNA head from scratch with the binary cross entropy loss with only 1 $Conv1D$ layer without OOV (out of vocabulary) token because we do not need to force the start and end position to learn OOV when we only predict answer or no-answer. However, this training setup did not work well, and only achieved the $AvNA$ score of 59.44. Further study is required to understand why this setup failed. One issue here is that because our handling of OOV is
changed, we can no longer finetune based on QANet which assumes the OOV token is the prefix for every training question and context.

**AvNA Head Small:** If we keep the OOV token, with the same AvNA Head consist of 1 Conv1D layer, we can finetune the model to achieve respectable score of 73.38 for AvNA. Comparing to our best model with 3 Conv1D layers for the AvNA Head, the larger model with decreasing channel size captures the local context of no-answer better.

**QANet Pred Head:** To predict no-answer, the original QANet uses $p_1(0) \times p_2(0)$ as the no-answer probability, where $p_1(0)$ and $p_2(0)$ means the probility of selecting OOV token at 0th position for both start and end position. We can take the same design and use $p_1(0) \times p_2(0)$ as our AvNA head, instead of producing $p_3$ and train it using same binary classification objective. Training from scratch did not yield meaning results with AvNA score of 59.67. Further study is required to determine if we can finetune this alternative AvNA design to work better.

**QANet AvNA Joint Training:** we experimented with training QANet and our default AvNA head with 3 Conv1d layers jointly using equation (3) from scratch. We did not fully explores the $\lambda$ parameter because each change require re-training the whole model. Our preliminary results show that a smaller $\lambda = 0.2$ produces the best results of 75.3 AvNA score comparing to larger parameters such as $\lambda = 2.0$. We hypothesize that because the original QANet training is already penalizing the no-answer case using start and end position prediction of OOV token, we can only put a small weight on our AvNA head objective during training.

![QANET ABLATION STUDIES](Figure 5: QANet ablation study of how different components can affect the final QA scores.)

![Comparison of Different AvNA Design](Figure 6: Comparison of different AvNA design. This study uses character embedding 64 instead of 200.)
6 Analysis

6.1 Positive Example

The SQuAD 2.0 dataset contains many questions and context that are adversarially crowdsourced which contributed the low performance of our baseline model. Table 2 shows the same question from our Introduction section and its corresponding context in SQuAD 2.0. The question asks about crop in Japan but the context is about China. The BiDAF baseline incorrectly extracted sorghum, which is indeed a crop brought to China, and treated it as the answer for Japan. By learning the local context using convolution, capturing longer word to word and character to character context using self-attention, and explicitly selecting answer or no-answer pairs using AvNA head, our model correctly predicts no-answer in this case.

<table>
<thead>
<tr>
<th>Question</th>
<th>What major crop was brought to Japan from the west?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>...Contacts with the West also brought the introduction to China of a major food crop, sorghum, along with other foreign food products and methods of preparation.</td>
</tr>
<tr>
<td>Baseline BiDAF Prediction</td>
<td>sorghum</td>
</tr>
<tr>
<td>Our QANANET Prediction</td>
<td>N/A</td>
</tr>
<tr>
<td>Golden Label</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 2: Positive example of our model correctly predicting no-answer for a hard question with adversarial context.

6.2 Negative Example

Our model is not perfect and Table 3 shows an example of model failing to predict no-answer. Our model only understands the context of number of Frenchmen in battle and selected the number of Frenchmen who joined the battle. However, our model does not infer the notion of lost, or losing people, especially when the "lost" keyword is not in the context. Solving this problem may require pre-training with larger English corpus to understand English at deeper level and the larger dataset might require larger transformer models such as BERT [7].

<table>
<thead>
<tr>
<th>Question</th>
<th>How many Frenchmen lost Battle of Carillon?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>The third invasion was stopped with the improbable French victory in the Battle of Carillon, in which 3,600 Frenchmen famously and decisively defeated Abercrombie’s force of 18,000 regulars...</td>
</tr>
<tr>
<td>Our QANANET Prediction</td>
<td>3,600</td>
</tr>
<tr>
<td>Golden Label</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 3: Negative example of our model incorrectly predicting answer for a hard question with adversarial context.

7 Conclusion

In this report, we use QANet as the solution to significant improve the baseline BiDAF model $F_1$ dev score from 60.71 to 69.34, and $EM$ from 57.10 to 65.52. We propose a new AvNA classification head to predict the probably of no-answer explicitly with a new binary cross entropy training objective and further improves the dev score to 70.37 for $F_1$ and 66.85 for $EM$. On the test leaderboard, our model achieves relatively high $F_1$ of 66.581 and $EM$ of 62.975.

As future work, we will complete our study on various AvNA head design and attempt to handle the failure case we show using more data and larger models. In addition, we will evaluate our AvNA head design as extension to other state-of-the-art models for SQuAD 2.0 such as BERT.
hypothesize that our AvNA head design and objective function can generalize to larger models to improve handling of the unanswerable questions.

References


A Appendix

Here is a summary of differences between QANet and BiDAF:

- **Input embedding layer** QANet uses trainable character embedding of 200 dimension, instead of the 64 dimension character embedding. We will show in the Experiments section that 200 dimension character embedding can outperform 64 dimension. And QANet uses hidden size of 128 instead of 100 for embedding of each question and context token.

- **Encoder layer** QANet replaces linear layers and RNN with encoder block of three basic operations: repeated convolution layer (depthwise separable convolution), self-attention layer, and feedforward layer for both question embedding and context embedding inputs. It is not clear from the QANet paper how positional embedding is combined with the original word and character embedding. This leaves some rooms for experimentation which we will address in the experiment section.

- **Question-context attention layer** QANet only uses context-to-query attention while BiDAF uses both context-to-query and query-to-context attention.

- **Model encoder layer** QANet replacing this layer with 3 stages of 7 encoder blocks with operations described in encoder layer. All three stages are sequentially connected with the first stage outputing $M_0$, second stage outputing $M_1$, and the third stage outputing $M_3$. All 3 stages share the same weights.

- **Output layer** no longer requires RNN. Instead, QANet concatenates $M_0$ with $M_1$, project them, and produce softmax probability of starting position $p_1$. The QANet uses similar operation for $M_0$ concatenated with $M_2$ for ending position $p_2$. 

10