Extending QANet for SQuAD 2.0

Stanford CS224N Default Project

Eun Jee Sung
Department of Computer Science
Stanford University
ejsung@stanford.edu

Ofure Ebhomielen
Department of Computer Science
Stanford University
ofure@stanford.edu

Abstract

In this paper, we extend the Question Answering Network (QANet) model for the question-answering task and train on a modified SQuAD 2.0 dataset. Our goal is to build a question answering model that can accurately predict whether a question is answerable, and if it is, where the answer span lies within the given context. To achieve this goal, we change the context-query attention mechanism of the basic QANet to a multi-headed context-query attention, capturing different interactions between context and query paragraphs. We also explore verifier models to output a binary prediction of whether a question is answerable or not. Finally, we experiment with various answer pointer networks. Our best ensemble model achieved F1 score of 69.57 and EM of 66.54 on the `test` dataset.

1 Key Information to include

- Mentor: Kendrick Shen
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

Question Answering (QA) is a task of predicting an answer to a given problem expressed in natural language. In the context of the Stanford Question Answering Dataset (SQuAD) challenge, we are given a context and question text, and our goal is to find a span of words in the given context paragraph that can answer the given question (Table 1). Question answering systems can be used in various domains, such as search engines and customer services.

Automated question answering with machine reading comprehension is challenging for several reasons. First, answering questions based on a paragraph requires understanding of long sequential data. Also, after understanding the given context and question paragraph, a model has to understand the relationship between the context and question paragraphs to find the final answer. Finally, recurrent neural network (RNN) structures that were traditionally used for the QA task required long training time due to their sequential nature.

In 2018, the attention-based methods led to exciting breakthroughs in developing question-answering deep learning models. QANet was among the first to apply the Transformer architecture to the QA domain. By adopting the Transformer architecture and removing RNN counterparts, they could achieve better performance on the official SQuAD leaderboard while reducing the training and inference time significantly by parallelizing the training process.

Inspired by the performance of QANet and recent trend in the neural language models using attention-based pre-trained language models, we decided to build a model based on QANet that is better suited for SQuAD 2.0 challenge. Unlike SQuAD 1.0, the SQuAD 2.0 dataset contains unanswerable questions, so models have to first determine if the given question is answerable based on the given context or not. Thus, we focused on improving the QANet with following approaches.

Context	Some disagree with such double or triple non-French linguistic origins, arguing that for the word to have spread into common use in France, it must have originated in the French language. The "Hugues hypothesis" argues that the name was derived by association with Hugues Capet, king of France, who reigned long before the Reformation. He was regarded by the Gallicans and Protestants as a noble man who respected people's dignity and lives. Janet Gray and other supporters of the hypothesis suggest that the name huguenote would be roughly equivalent to little Hugos, or those who want Hugo.
Question	Who was the first king of France to reign during the Reformation?
Answer	Not Answerable

Table 1: An example of the SQuAD dataset and its model prediction.

- We implemented a multihead context-query attention layer.
- We built an answerability verifier module predicting whether a question is answerable.
- We experimented with various answer span pointer networks and losses.

3 Related Work

3.1 Question Answering

Answering questions using a given context requires building accurate representations of question, context, and the interaction between them. Our default baseline model, the Bi-Directional Attention Flow (BiDAF) model uses Bi-directional Long Short-Term Memory (BiLSTM) model and attention mechanism [1]. Its main mechanism is the bi-directional query-to-context and context-to-query attention flow, which flows between each step of the LSTM and helps the model to build the context representation conditioned on query.

Following the proposal of the Transformer, several models have adopted the Transformer and self-attention mechanisms. QANet [2] (Fig. 1) adopts the Transformer architecture to remove recurrent mechanisms inside the encoders. It takes the character and word-level embeddings of context and answer paragraphs and outputs the starting and ending boundary of the answer span within the context paragraph. Removing recurrent structures reduces its training time significantly by parallelizing the sequential training process inherently required for sequential models.

More recent state-of-the-art models use pre-trained language models such as BERT [3], AlBERT [4], and RoBERTa [5]. Such systems concatenate embeddings from pre-trained language models to basic word- and character-level embeddings to take the advantage of well-defined embeddings supported by the vast amount of language dataset learned by the models. However, because we were prohibited from using pretrained models or pre-implemented model codes, we could not use such approaches.

3.2 Answer Pointer Network

Pointer Networks, as described by the original paper [6], are neural architectures that learn the conditional probability of an output sequence with discrete tokens corresponding to positions in the input sequence. The novelty in pointer networks is that attention is now used as a pointer to select a member of the input sequence as the output, rather than using attention to blend hidden units of an encoder to a context vector at each decoder step.

Several QA models have adopted Pointer Networks to improve the accuracy of their models. [7] used the hidden state of the answer start token to predict the answer end token, in the prediction layer of their model. [8] proposed a framework named CARTON (Context Transformer Stacked Pointer Networks), which was able to perform multi-task semantic parsing and efficiently handles the problem of conversational QA over a large scale knowledge graph. Their framework used a stack of pointer networks to parse the input question and the dialog history and then generated a sequence of actions that can be executed on the knowledge graph. [9] used answer pointer network that detected the answer boundaries from the passage when the question is answerable with two trainable matrices

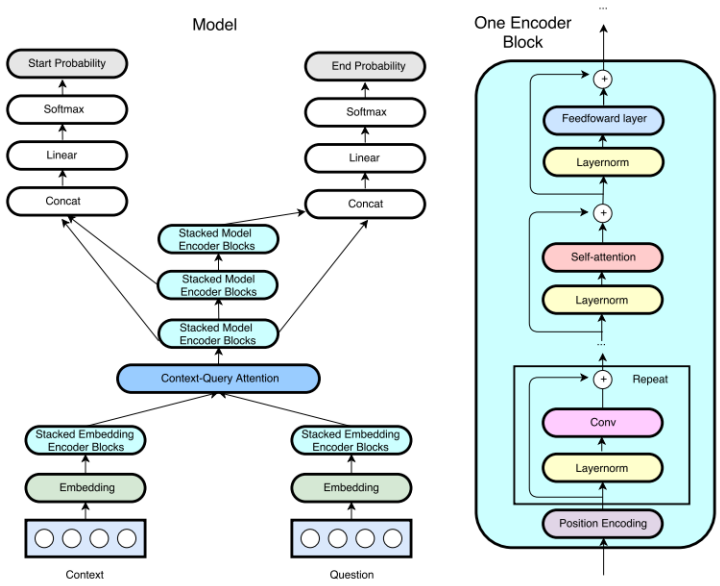

Figure 1: QANet model architecture, taken from the original paper [2]

W_s and W_e to estimate the probability of the answer start and end boundaries of the ith word in the passage, α_i and β_i. In fact, boundary-predicting approach is widely adopted by QA models [1, 2].

4 Approach

4.1 Baselines

The baseline for this project is the BiDAF implementation which was provided as a starter code by the course staff. As stated in 3.1., the BiDAF model is a multi-stage hierarchical model representing the context at different granularity levels, and further uses bi-directional attention flow mechanism to obtain a query-aware context representation.

4.2 Question-Answering Network (QANet)

QANet has five major parts including the initial embedding, embedding encoding, and context-query attention layer (Figure 1). We describe each part below.

Input Embedding Layer. The input embedding layer receives the fixed pretrained GLoVe word embedding of 300-D and trainable character embedding of 200-D. We then pass the concatenated embedding of dimension 500 to a two-layer highway network. We reuse the character embedding and highway network implementations from the BiDAF model implementation by the course staff. We use hidden dimension size of 256.

Embedding Encoder Layer. We stack multiple encoding blocks for the embedding encoder layer, whose weights are shared between context and query embeddings. An encoding block consists of multiple layers of the depthwise separable convolutions [10], a self-attention, and a final feed-forward layer and uses the hidden dimension of 256.

(Multihead) Context-Query Attention. Based on the equivalence of equations for the bidirectional context-query attention, we simply reuse the BiDAF model's bidirectional attention module for the default QANet implementation.

We then extend the structure to a multihead version to motivate different heads to focus on different types of context-query interaction (Figure 2). We use hidden dimension of 256 separated into 8 heads for this layer. We found that the multihead context-query attention indeed improves the model performance (Table 2).

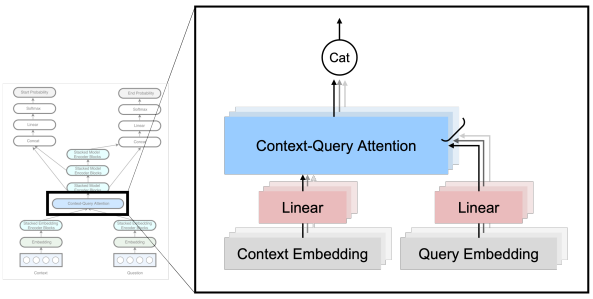

Figure 2: We replaced the context-query attention layer with a multiheaded variant.

Model Encoder Layer. This layer takes in the concatenated bidirectional attention scores and encodes it to the hidden embeddings to predict the starting and ending position. There are three model encoder layers sharing the same model weight. Note that the original QANet uses seven encoder blocks inside the model encoder, but we had to decrease it to five layers due to the limitation in computing power.

Output Layer. The original QANet uses a simple single-layer model for starting and ending point prediction. Specifically, it applies a fully-connected layer to the concatenated feature maps from the first and second model encoder sublayer to predict the starting position, and applies another fully-connected layer to the feature maps from the first and third sublayer to predict the ending position. We discuss our improvements over this default structure in detail in 4.3.

We implemented the QANet from scratch without reference to existing implementations other than the BiDAF model, course assignment codes, and PyTorch APIs in accordance with the honor code.

4.3 Improving the Output Layer

The original QANet only uses single linear layer for each starting and ending boundary position prediction (Figure 3(d)), i.e.

$$p^1 = softmax(W_1[M_0; M_1]), \quad p^2 = softmax(W_2[M_0; M_2]) \tag{1}$$

where p^1 and p^2 are respectively the answer span's starting and ending boundary prediction, W_1 and W_2 are two trainable variables, M_i is the output from the i-th model encoder layer [2].

We focused on changing the final output layers of the QANet to improve model accuracy. We experimented with various additional auxiliary losses to improve our answer pointer layer. These modules can be either used as auxiliary output nodes to guide the boundary prediction layer's training with additional losses and/or replace the simple boundary predictor. We describe each module below and visualize them in Figure 3.

Answerability Verifier (V). For the SQuAD 2.0 challenge, we have to predict whether a question is answerable or not. Thus, inspired by the verifiers in [11, 12, 9], we implemented a simple verifier module for the task, which produces a binary output if a given question is answerable. This can be expressed as below, where $M_2[0, :]$ denotes the feature map from the 0-th index (which denotes the 'unanswerable' token) of M_2.

$$pv_i = sigmoid(W_v M_2[0, :]) \tag{2}$$

To train the module, we use the binary cross-entropy loss between the model output probability and the target answerability (0 = unanswerable, 1 = answerable) (Figure 3(a)), i.e.

$$L_v(\theta) = -\frac{1}{N}\sum_{i}^{N}[yv_i \times \log pv_i + (1 - yv_i) \times \log(1 - pv_i)] \tag{3}$$

where yv_i a binary indicator whether the given question i is answerable and pv_i is the predicted possibility of the question i being answerable. When using this module, we zeroed out the answer span predictor's boundary prediction and treated a question as unanswerable if the output of this module was below certian threshold.

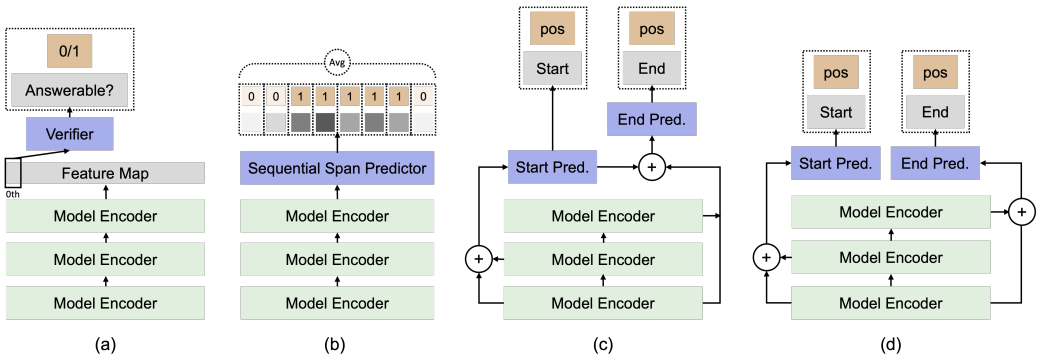

Figure 3: Variations of answer pointer model. (a) Answer verifier module. (b) Sequential answer pointer module. (c) Answer boundary pointers where the ending point predictor is conditioned on the starting point prediction. (d) Answer boundary pointers suggested by the original QANet. Our best model uses (a), (b), and (d) pointers at the same time (Appendix A).

Sequential Answer Pointer (SEQ). This mechanism was inspired by the sequential model from [13]. We added a node predicting whether a word was in the answer span or not (Figure 3(b)). We expected that this auxiliary node would provide more information than the errors from the starting and ending point prediction, since a model might be wrong by a few words, but that doesn't necessarily mean that the model is attending to the wrong part of the paragraph. It can be expressed as the equation below, where s denotes the context paragraph and W_v is a trainable parameter of shape $(d, 1)$. p_s is a vector of shape (sequence length, 1) where each value indicates the probability of the corresponding token included in the answer span.

$$p_s = sigmoid(W_v M_0) \tag{4}$$

This module can be trained with the loss term below.

$$L_s(\theta) = -\frac{1}{N}\sum_{i}^{N}[\frac{1}{|S_i|}\sum_{c \in S_i} y_c \times \log p_c + (1 - y_c) \times \log(1 - p_c)] \tag{5}$$

where S_i is the token sequence of the given context i, c is a token in S_i, y_c a binary indicator whether token c is included in the answer span or not, and p_c is the predicted possibility of token c being in the answer span. In other words, this loss corresponds to the mean binary cross-entropy loss of predictions for all tokens in our training set.

Conditioning Ending Point Pointer (COND). As reported by [13], we also wanted to test if the accuracy would be improved when the end pointer is conditioned on the start pointer. The goal is to calibrate the end pointer so that its output index is always after the start index as we thought this would give a more accurate answer boundary. We experimented with two single-layer feed-forward networks for each of the start and end prediction pointers (Figure 3(c)).

$$p^1 = softmax(W_2 \times relu(W_1[M_0; M_1])), \quad p^2 = softmax(W_4 \times relu(W_3[M_0; M_2; p^1])) \tag{6}$$

where p^1 and p^2 are respectively the answer span's starting and ending boundary prediction, W_1, W_2, W_3, W_4 are trainable variables, M_i is the output from the i-th model encoder sublayer.

This module can be trained using the same loss function as the original QANet boundary prediction model (Eq. 7).

Ensemble. For the in-class leaderboard, we ensembled the results from four model variants using majority-vote rule, namely the "QANet," "QANet + V + SEQ," "Multihead QANet + V," and "Multihead QANet + V + SEQ" models in Table 2. We picked the combination of model variants that empirically performed the best on the `dev` leaderboard. We chose the result from the "Multihead QANet + V + SEQ" model whenever tie occurred.

4.4 Loss Function

To train the final layer variants suggested above, we add the corresponding auxiliary losses to the original loss to calculate the final loss.

The original QANet's loss function is a sum of negative log probabilities of predicted starting and ending points, indexed by true start and end indices. Concretely,

$$L_{qanet}(\theta) = -\frac{1}{N}\sum_{i}^{N}[\log(p^1_{y^1_i}) + \log(p^2_{y^2_i})] \tag{7}$$

where y^1_i and y^2_i are respecively the groundtruth starting and ending position of the example i, and θ is the set of all trainable variables.

Our best model uses the verifier (V) and sequential prediction model (SEQ) on top of the multihead QANet model (Appendix A). Thus, our best model's loss function is as follows.

$$L_{best}(\theta) = L_{qanet}(\theta) + L_s(\theta) + L_v(\theta) \tag{8}$$

4.5 Failed Experiments

Although they went in vain, we tried other variations of QANet. Here we list up some of the ideas we had to early-stop due to poor performance or instability during the training.

- Training the context and query encoder separately: In our final model, context and query paragraph encoders share the weights. Using separate encoders worsened performance, and we suspect sharing weights allows the encoder to observe more training examples and help align the context and query distribution.
- Local-aware feed-forward network using 1-D convolution: We tried replacing the feed-forward layer in the encoder block with 1-D convolution with fixed kernel size to better capture locality of tokens.
- Using the maximum value of probability prediction to predict answerability: We tried to calibrate answerability prediction by approximating confidence level of answer span prediction with the maximum value of the final softmax output.

5 Experiments

5.1 Data

We use the modified SQuAD 2.0 dataset offered by the course staff as a part of the Default Project - I.I.D. SQuAD track [1]. As stated in the project guideline, it contains 129,941 `train`, 6,078 `dev`, and 5,915 `test` examples [2]. We did not use any other external dataset, including the original SQuAD dataset [14].

5.2 Evaluation Method

We follow the class SQuAD leaderboard and use Exact Match (EM) and F1 score as our evaluation metrics. We include the average answerability prediction accuracy (AvNA) as our third metric. Exact Match is a binary measure (0 or 1) strictly reporting whether the system's output is an exact match with the ground truth answer. F1 score is the harmonic mean of the precision and recall. AnVA is the average recall of non-answerable questions.

5.3 Experimental Details

We trained the baseline and experimental models using the whole training set. The BiDAF model was trained only using word embeddings. Following the original paper, the QANet and its variants were trained using both the word and character embeddings.

We trained the models using the default pipeline provided by the course staff. For the BiDAF baseline model, we used the default set of hyperparameters provided in the `args.py` file. For the QANet-based models, we mostly followed the set of hyperparameters specified in the paper including the learning

[1] https://github.com/michiyasunaga/squad

[2] http://web.stanford.edu/class/cs224n/project/default-final-project-handout-squad-track.pdf

rate warmup process during the first 1000 steps, after which the learning rate stayed constant as 0.001. We also applied dropout layers inside encoder blocks, but with the dropout rate of 0.2 instead of 0.1 in the paper and applied the standard L2 weight decay with $\lambda = 3 \times 10^{-7}$. We used the Adam optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. For all models, we used batch size of 32 and trained for 30 epochs. We used one NVIDIA V100 GPU to train all models. We used the `pytorch_warmup` `pip` package [3] for the warmup, but didn't use any other pre-implemented code repositories.

While producing the final prediction output, we added a simple rule-based post-processing for the model variants with verifier modules. We used the output of the verifier module to predict if a given question was answerable and ignored the starting and ending position prediction to mark the question 'unanswerable' if it was less than the fixed threshold (0.7).

5.4 Results and Ablations

Model	Embedding	AvNA	F1	EM
BiDAF (Baseline)	W	68.43	61.46	58.12
QANet (a)	W + C	68.66	61.57	57.97
QANet + V	W + C	71.16	63.96	60.28
QANet + V + SEQ (b)	W + C	**71.84**	**65.01**	**61.12**
Multihead QANet + V (c)	W + C	72.06	65.74	62.16
Multihead QANet + SEQ	W + C	**72.71**	66.4	62.85
Multihead QANet + COND	W + C	62.44	50.91	47.16
Multihead QANet + V + SEQ (d)	W + C	72.68	**66.88**	**63.38**
Multihead QANet + V + SEQ + COND	W + C	70.70	63.57	59.84
Ensemble (a+b+c+d)	W + C	**74.48**	**69.57**	**66.54**

Table 2: Preliminary result of the model evaluated on the `dev` set. In the 'Embedding' column, 'W' refers to word-level and 'W+C' refers to the word and character-level embeddings.

The results of our model variants on the SQuAD 2.0 `dev` set are shown in Table 2. Note that due to limitations in computing resource, we could not perform rigorous ablations across the entire combinations of modules, but we report as many ablations as possible under our constraints, especially among the multihead model variants.

Based on the overall F1 and EM score, our best model was "Multihead QANet + V + SEQ," using the baseline QANet model, answerability verifier module, and the sequential predictor module together. Our extended model achieved superior performance to the baseline BiDAF model with AvNA of 72.68, F1 of 66.88, and EM of 63.38 on the `dev` dataset. The ensemble model of four model variants, achieved significant gain over the baseline model with AvNA of 74.48, F1 score of 69.57, and EM of 66.54 on the `dev` leaderboard. It achieved F1 score of 66.3t and EM of 63.25 on the `test` leaderboard.

We saw increase in model performance when we used the verifier and sequential prediction model. We saw slightly better performance when we used the sequential prediction model than the basic QANet or QANet and the verifier module, confirming that the sequential auxiliary loss helped in model training. Interestingly, conditioning ending predictor on the starting boundary prediction did not improve performance. We observed the training and validation loss increase after getting to a certain local minimum during training, alongside some fluctuations with the F1 and EM scores. We suspect that we might have needed to tune the hyperparameters (e.g. the learning rate) to accommodate the additional layers to the output.

6 Analysis

6.1 Answerability Prediction

For the verifier module, we first used threshold of 0.5 but ultimately adjusted it to 0.7 after witnessing that the 0.5 threshold resulted in high false positive rate. We generally see high true negative and

[3] https://pypi.org/project/pytorch-warmup/

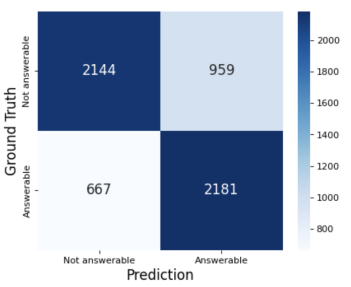

Figure 4: Confusion matrix of answerability prediction

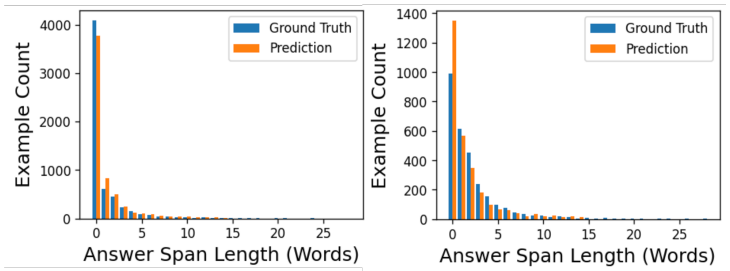

Figure 5: Length distribution of ground truth and predicted answer spans. Left: Answer span lengths of all questions. Right: Answer span lengths of answerable questions.

positive rate on the textttdev dataset, with the average AvNA of 72.71%, but we find that further improvements can be made to reduce the false positive rate (Figure 4).

Among the false positive predictions, we observe that the model is prone to output one-word questions to unanswerable questions (Figure 5). Most likely, the model tries to output at least the one word with the highest softmax probability while trying to predict the answer span. Qualitatively, we observed that the model either outputted entirely wrong answer or a characteristic word included in both the query and context paragraph for one-word answers.

6.2 Qualitative Error Analysis

To examine where our model was suffering, we performed a qualitative evaluation on individual failure examples. First, we noticed that the model failed to produce concise answers. For example, it returned "3 miles (5 km) west-northwest" to the question, "How far from state house in downtown Boston is Harvard Yard?" when the ground truth was "3 miles." We also noticed that the model suffered to detect unanswerable questions when the question and candidate context span shared similar tokens, but in fact one token made significant change in the meanings. For example, for the question "In a 4-cylinder compound engine, what degree were the individual cranks balanced at?" on the context "(omitted) the individual pistons within the group are usually balanced at 180° (omitted)," the model outputted "180°" as the answer when in fact, the question about cranks was not answerable from the given context on pistons. Finally, the model had problem capturing answer spans of complex grammatical structure, such as outputting only "Iroquois Six Nations" when the ground truth was "Iroquois Six Nations, and also by the Cherokee."

7 Conclusion

In this paper, we introduce our attempts to implement and extend the QANet model for the modified SQuAD 2.0 dataset. We use Transformer- and convolution-based QANet to encode question and context paragraphs, and adopt multihead bi-directional attention to better capture the similarity between questions and contexts. Our experiment results show that our model can better predict the answerability of questions and answer spans for answerable questions than the baseline BiDAF or original QANet model. Our basic ablation study demonstrates that the answerability verifier and sequential prediction loss can help improve the prediction.

Future works include reducing false positive predictions, namely false-positive one-word predictions. It could possibly be done by explicitly putting the verifier in front of answer span boundary prediction, and developing more powerful verifiers. Also, we could potentially capture the context-query attention in a more fine-grained way by making context-query multi-hop or multi-scale.

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] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] 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.

[5] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[6] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

[7] Marshall Ho, Zhipeng Zhou, and Judith He. When to fold'em: How to answer unanswerable questions. *arXiv preprint arXiv:2105.00328*, 2021.

[8] Joan Plepi, Endri Kacupaj, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. Context transformer with stacked pointer networks for conversational question answering over knowledge graphs. In *European Semantic Web Conference*, pages 356–371. Springer, 2021.

[9] Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. U-net: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1810.06638*, 2018.

[10] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[11] Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6529–6537, 2019.

[12] Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, GP Bhargav, Dinesh Garg, and Avirup Sil. Span selection pre-training for question answering. *arXiv preprint arXiv:1909.04120*, 2019.

[13] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016.

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

A Appendix: Final Model Output Layer

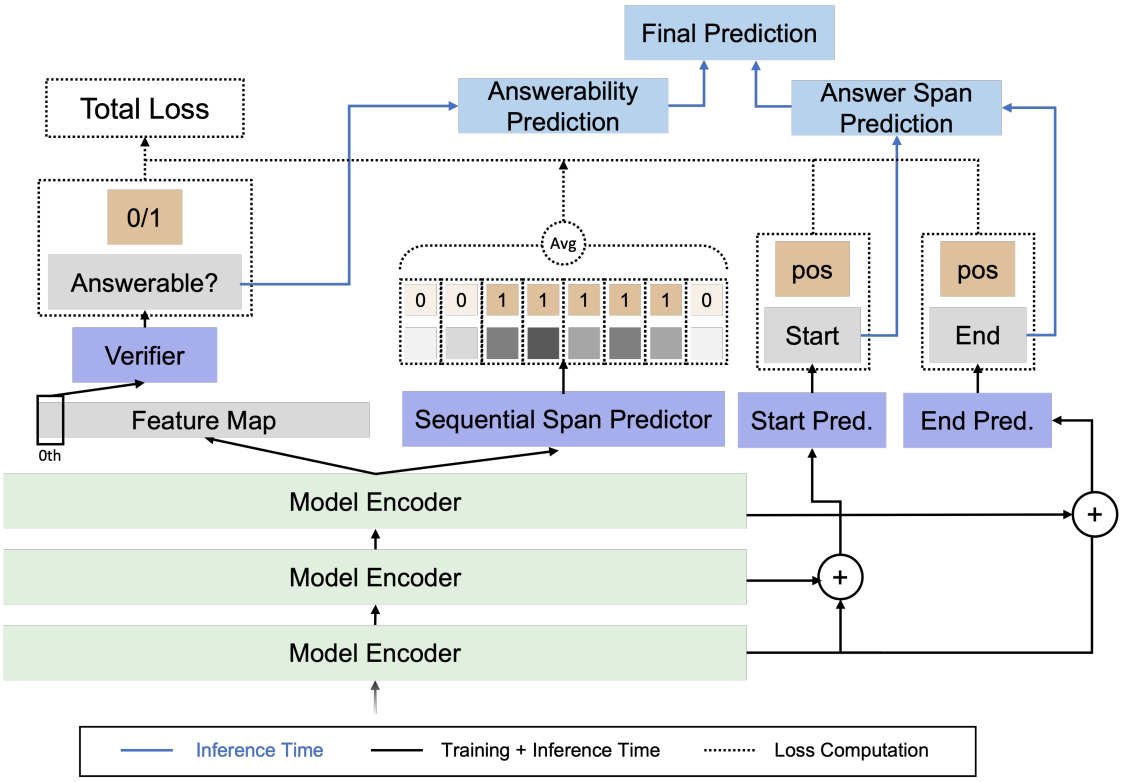

Figure 6: Final output layer of our best model