Wisdom of Ignorance: BERT-wBiDAF for SQuAD2

Siyi Tang (tsy935@stanford.edu),
Ruge Zhao (rugezhao@stanford.edu),
Meixian Zhu (mxzhu@stanford.edu)

Abstract

SQuAD v2.0, as compared to SQuAD v1.0, has a major change in including unanswerable questions into the dataset. This has led to significant performance drop in models previously trained for SQuAD v1.0. We propose two main areas in improving the state-of-the-art BERT model for SQuAD v2.0. First, to improve the model for correctly distinguishing between answerable and unanswerable questions, we take answerable/unanswerable classification into account either through a two-stage model or adding an extra binary cross-entropy loss into the training objective. Secondly, we believe that an answer is closely related to the type of the question. We encourage the model to focus on question types by either training a model on each type separately or introducing a weighted attention mechanism. The best model is a combination of BERT pre-trained embeddings and BiDAF architecture with classification loss (BERT-wBiDAF), where the context-to-question attention is up-weighted for keywords in the questions. Our BERT-wBiDAF model achieves 73.89 EM, 77.45 F1 and 81.11 AvNA on the dev set, and 72.97 EM and 76.66 F1 on the test set.

1 Introduction

Question answering (QA) system is designed to automatically answer questions posed by humans in a natural language. Question answering tasks are challenging, because it requires both understanding of natural language and knowledge about the world. Several open datasets have been collected to advance research in question answering, such as the Stanford Question Answering Dataset (SQuAD) v1.1 [Rajpurkar et al., 2016] and v2.0 [Rajpurkar et al., 2018], TriviaQA [Joshi et al., 2017] and NewsQA [Trischler et al., 2016].

State-of-the-art question answering models can be divided into two main types: models with pre-trained contextual embeddings (PCE) and models without PCE. In the non-PCE division, one representative is the Bi-Directional Attention Flow (BiDAF) network. BiDAF is a hierarchical multi-stage architecture that uses bi-directional attention flow to obtain a query-aware context representation [Seo et al., 2016]. Whereas in the PCE division, BERT is one of the most representative models. It is a language representation model designed to pre-train deep bidirectional representations [Devlin et al., 2018]. The pre-trained BERT representations can be easily fine-tuned for a wide range of natural language tasks such as question answering, sequence classification and named entity recognition. BERT fine-tuned for question answering achieves 93.2 test F1 on SQuAD v1.1, outperforming human performance by 2.0. However, test F1 score obtained by BERT drops by 10 on SQuAD v2.0 (https://rajpurkar.github.io/SQuAD-explorer/).

In this paper, we propose two main areas in improving BERT for SQuAD v2.0. First, to improve the model for correctly distinguishing between answerable and unanswerable questions, we take answerable/unanswerable classification into account either through a two-stage model or adding an extra binary cross-entropy (BCE) loss for answerable/unanswerable classification. Secondly, we encourage the model to focus on question types by either training a model on each type separately or combining pre-trained BERT representation with BiDAF architecture using a weighted attention mechanism. Experiments showed that BERT-wBiDAF with BCE loss (BERT-wBiDAF in short)

has the best performance on dev set, with 73.89 EM, 77.45 F1 and 81.11 AvNA. On test set, our BERT-wBiDAF model achieves 72.97 EM and 76.66 F1.

2 Related Work

Most of the successful models for question answering usually involve a recurrent neural network to process sequential inputs as well as attention to capture long-term interactions. One example of such combination is BiDAF [Seo et al., 2016], which includes a bi-directional attention both from the context to the question and from the question to the context. In contrast, QANet [Yu et al., 2018] aims to tackle the low training efficiency involved in training a recurrent network, and exclusively uses only convolutions and self-attentions to encode context and query separately.

BERT [Devlin et al., 2018] is the current state-of-the-art pre-trained contextual representation. BERT can be fine-tuned easily and inexpensively for many natural language processing tasks (e.g., question-answering, natural language understanding etc).

ELMo [Peters et al., 2018] trains left-to-right (LTR) and right-to-left (RTL) models separately and represents each token as a concatenation of the two models. However, BERT is claimed to be better than ELMo in the following ways: (a) ELMo is twice as expensive as a single bidirectional model; (b) ELMo is non-intuitive for tasks like question answering, since the RTL model would not be able to condition the answer on the question; (c) ELMo is strictly less powerful than a deep bidirectional model, since a deep bidirectional model could choose to use either left or right context [Devlin et al., 2018].

OpenAI GPT [Radford et al., 2018] is the closest pre-training method to BERT. The major improvement in terms of methodology in BERT arises from the two pre-training methods: Masked LM and Next Sentence Prediction. In addition, GPT only uses sentence separator [SEP] and classifier token [CLS] during fine-tuning, while BERT learns both tokens as well as the sentence A/B embeddings during pre-training stage.

3 Approach

3.1 Baselines

3.1.1 BiDAF Baseline

We used the BiDAF baseline [Seo et al., 2016] provided by the teaching team as the weaker baseline of our project.

3.1.2 BERT Baseline

In addition to BiDAF, we fine-tuned pre-trained BERT embeddings (base, uncased) as the stronger baseline. To fine-tune BERT for question answering task, we followed the same procedure as recommended by [Devlin et al., 2018]. We built upon the code implementation from Hugging Face (`https://github.com/huggingface/pytorch-pretrained-BERT`).

In particular, we added a linear layer to BERT to predict the start and end positions of the answer span. The only new parameters learned during this fine-tuning stage are a start vector $S \in \mathbb{R}^H$ and an end vector $E \in \mathbb{R}^H$. Suppose the final hidden vector from BERT for the i^{th} input token be denoted as $T_i \in \mathbb{R}^H$. Then, the probability of word i being the start of the answer span is computed as a dot product between T_i and S followed by a softmax over all of the words in the paragraph: $P_i = \frac{e^{S \cdot T_i}}{\Sigma_j e^{S \cdot T_j}}$. The same formula is used for the end position of the answer span, and the maximum scoring span is used as the final prediction.

3.2 Our Approach

We propose two main areas to improve BERT on SQuAD v2.0. First, we believe that one of the challenges in SQuAD v2.0 is the unanswerable questions. Hence, we hypothesize that incorporating classification of answerable/unanswerable questions into our model would boost the overall performance. We experimented in two ways to incorporate classification: (a) training a classifier specifically

for answerable/unanswerable classification (section 3.2.1); (b) adding the classification loss into the total loss (section 3.2.4).

Furthermore, we observed that questions in SQuAD v2.0 can be categorized into several types based on keywords. Hence, we experimented with two ways of encouraging the model to focus on different question types by (a) training a model for each question type separately (section 3.2.2); (b) combing BERT and BiDAF architecture with extra attention weight given to keywords (sections 3.2.3 and 3.2.4).

Figure 1 shows an overview of our approaches. The details of each approach is explained in the following subsections.

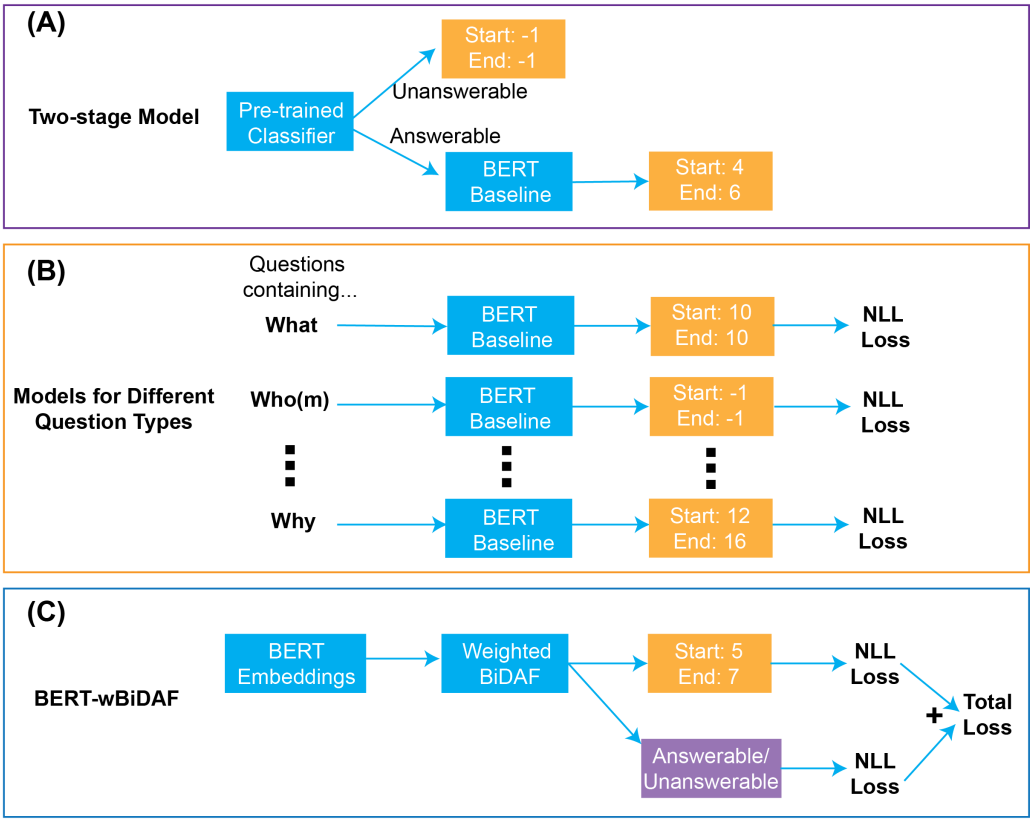


Figure 1: Overview of Our Approaches

3.2.1 Two-stage Model: Classifier + BERT Baseline

As a preliminary experiment, we ran the BERT baseline on answerable training examples only (similar to SQuAD v1.0), and observed significant improvement as compared to the same model trained on all training examples. This observation motivates us to build a strong classifier to filter out unanswerable questions. Hence, we propose a two-stage model here (Figure 1A). The first stage is a binary classifier for answerable/unanswerable prediction, and the second one is the BERT baseline model to predict the answer span.

We experimented with two architectures for the classifier. First, we followed the fine-tuning procedure for sentence classification in [Devlin et al., 2018]. More specifically, we use the final hidden vector $C \in \mathbb{R}^H$ corresponding to the first input token [CLS] (i.e., $C_{[CLS]}$) as the aggregate representation for classification. We then pass $C_{[CLS]}$ through a linear layer to obtain answerable/unanswerable logits.

Our second implementation of classifier is a single-layer convolutional neural network (CNN). Here, we pass $C_{[CLS]}$ through a convolutional layer, followed by ReLU activation and max-pooling over

all the words in the sequence. Finally, we apply dropout and a linear layer to obtain the logits for classification.

In both of the implementations of classifier, we use binary cross-entropy (BCE) loss for training.

At prediction time, only the answerable questions predicted by the classifier are passed down to the second stage model (BERT baseline) for predicting the start and end positions of the answer span.

3.2.2 BERT Trained on Different Question Types

After pre-processing the original question sequences, we noticed that most of the questions can be classified into seven main types based on the keyword in the question. The seven types are: *what*, *who(m)*, *how*, *which*, *when*, *why*, and *where*. We fine-tuned one BERT baseline model for each of the question type in the training set (Figure 1B). All questions that do not contain a keyword were assigned to *others* category. The intuition behind this setup is to encourage our models to focus more on semantics corresponding to each question type through separate fine-tuning on further divided sub-tasks.

3.2.3 BERT-weighted BiDAF

Inspired by the findings in section 3.2.2, we hypothesize that encouraging the model to focus more on the keywords in the question would improve the model performance. Hence, we propose to combine the architecture of BiDAF with BERT pre-trained embeddings, and up-weight the Context-to-Question attention of the keywords in the questions.

Let the final hidden states of the sequence (question and context) from BERT be $\mathbf{s_1}, ..., \mathbf{s_{N+M}} \in \mathbb{R}^H$. We first split $\mathbf{s_1}, ..., \mathbf{s_{M+N}}$ into hidden states for question $\mathbf{q_1}, ..., \mathbf{q_M} \in \mathbb{R}^H$ and context $\mathbf{c_1}, ..., \mathbf{c_N} \in \mathbb{R}^H$. Then we compute the similarity matrix $\mathbf{S} \in \mathbb{R}^{N \times M}$ as described in [Seo et al., 2016] and the default project handout.

Next, we perform Context-to-Question (C2Q) Attention with *extra weight* given to the *keywords* in the questions:

$$\bar{\mathbf{S}}_{i,:} = softmax(\mathbf{S}_{i,:}) \in \mathbb{R}^M, \forall i \in \{1, ..., N\}$$
$$\mathbf{a}_i = \sum_{j=1}^{M} \bar{\mathbf{S}}_{i,j} k_j \mathbf{q_j} \in \mathbb{R}^H, \forall i \in \{1, ..., N\}$$

where $k_j = k$ if q_j corresponds to a keyword, and $k_j = 1$ otherwise. Here, k is a tunable hyperparameter which controls the weight of the C2Q attention of keywords in the questions.

We compute Question-to-Context (Q2C) Attention the same way as that in the BiDAF baseline according to the following formula:

$$\bar{\bar{\mathbf{S}}}_{:,j} = softmax(\bar{\mathbf{S}}_{:,j}) \in \mathbb{R}^N, \forall j \in \{1, ..., M\}$$
$$\mathbf{S'} = \bar{\mathbf{S}}\bar{\bar{\mathbf{S}}}^T \in \mathbb{R}^{N \times N}$$
$$\mathbf{b}_i = \sum_{j=1}^{N} \mathbf{S'}_{i,j} \mathbf{c_j} \in \mathbb{R}^H, \forall i \in \{1, ..., N\}$$

Then for each context location $i \in \{1, ..., N\}$, the output of the attention layer is:

$$\mathbf{g}_i = [\mathbf{c}_i; \mathbf{a}_i; \mathbf{c}_i \circ \mathbf{a}_i; \mathbf{c}_i \circ \mathbf{b}_i] \in \mathbb{R}^{4H}, \forall i \in \{1, ..., N\}$$

Then the question and context vectors are further refined using the same architecture of modeling layer as the BiDAF baseline.

Finally, we apply the same BiDAF baseline output layer to the outputs to obtain probabilities for start and end positions of answer span, i.e. $\mathbf{p}_{\text{start}}$ and $\mathbf{p}_{\text{end}}$. Let the gold start and end positions be $i \in \{1, ..., N\}$ and $j \in \{1, ..., N\}$ respectively, then the loss function of a single example is:

$$\text{loss} = -\log \mathbf{p}_{\text{start}}(i) - \log \mathbf{p}_{\text{end}}(j)$$

The loss is then averaged over all the examples in a mini-batch.

3.2.4 BERT-weighted BiDAF with BCE Loss (BERT-wBiDAF)

In contrast to the two-stage model described in section 3.2.1, here we incorporate classification BCE loss into the training objective. The model architecture is exactly the same as that in section 3.2.3 except that we add an extra answerable/unanswerable BCE loss into the total loss function (Figure 1C).

More specifically, we apply a pooling layer to modeling layer outputs to extract only the representation for the first input token [CLS] as the aggregate representation. Concretely, let $\mathbf{m_1}, ..., \mathbf{m_N} \in \mathbb{R}^{2H}$ be the modeling layer outputs, and $M \in \mathbb{R}^{2H \times N}$ be the matrix with columns $\mathbf{m_1}, ..., \mathbf{m_N}$. Then the probabilities for answerable/unanswerable $\mathbf{p}_{cls}$ are computed as follows:

$$\mathbf{P} = \mathbf{M}_{:,1}$$
$$\mathbf{p_{cls}} = sigmoid(\mathbf{W_{cls}}\mathbf{P})$$

Finally, let the gold start and end positions be $i \in \{1, ..., N\}$ and $j \in \{1, ..., N\}$ respectively, and the gold answerable/unanswerable category be $y \in \{0, 1\}$, then the loss function of a single example is:

$$\text{loss}_{\text{QA}} = -\log \mathbf{p}_{\text{start}}(i) - \log \mathbf{p}_{\text{end}}(j)$$
$$\text{loss}_{\text{cls}} = -\log \mathbf{p}_{\text{cls}}(y)$$
$$\text{Total loss} = \text{loss}_{\text{QA}} + \gamma * \text{loss}_{\text{cls}}$$

where γ is a tunable hyperparameter controlling the effect of misclassification on answerable/unanswerable questions. The loss is averaged over all the examples in a mini-batch.

4 Experiments

4.1 Data

The main dataset we used is SQuAD v2.0 [Rajpurkar et al., 2018]. We also used adversarial dataset [Jia and Liang, 2017] to test the robustness of our model to distracting sentences.

4.2 Evaluation Methods

We used EM, F1 and AvNA scores on dev set as the main evaluation metrics.

4.3 Experiment Details

4.3.1 BERT Baseline

The model configuration and hyperparameters used in BERT baseline are listed in Table 1. Due to memory constraint, the batch at each step was further divided by 2 (gradient_accumulation_step), and the gradient was accumulated over 2 steps.

4.3.2 Classifier

The model configurations and hyperparameters used for the classifier were the same as the BERT baseline (see Table 1). For the CNN classifier, we experimented with kernel size 2, 5 and 10.

4.3.3 BERT Trained on Different Question Types

With the same hyperparameters as the BERT baseline model (see Table 1), we trained one model for each question type, using only the training examples belonging to that question type. At prediction time, we first categorized the questions into the seven categories based on keywords in the questions, and then used the corresponding trained model for prediction. As for examples belong to *others* in the dev set, we used the BERT baseline model (trained on all training examples) to make predictions.

Configuration	Value
bert_model	bert-base-uncased
train_batch_size	12
learning_rate	3e-05
num_train_epochs	2
max_seq_length	384
gradient_accumulation_steps	2
hidden_size	768
num_hidden_layers	12
num_attention_heads	12
intermediate_size	3072
hidden_activation	gelu
hidden_dropout_prob	0.1
attention_probs_dropout_prob	0.1
max_position_embeddings	512
optimizer	BERT version of Adam

Table 1: Experimental details for BERT baseline

Configuration	Value
γ for weighting effect of classification loss	0.25
k for weighting keyword attention	3.5

Table 2: Extra experimental details for BERT-wBiDAF

4.3.4 BERT-wBiDAF without and with BCE Loss

We used the same model configuration and hyperparameters listed in Table 1 to train BERT-weighted BiDAF models (without and with BCE loss). Due to resource constraint, we used BERT-base pre-trained embeddings rather than BERT-large embeddings. In addition, we experimented with different values of the keyword attention weight k and the classification loss scaling factor γ. The final values used are $k = 3.5$ and $\gamma = 0.25$. (Table 2).

5 Results

5.1 Answerable/Unanswerable Classification Task

The best classifier is the CNN classifier with kernel size 2, which achieved 0.84 precision, 0.65 recall and 0.73 F1 on the dev set using 0.5 as the threshold. This result suggests that among all the questions identified as unanswerable, most of them are truly unanswerable. However, the model failed to detect all or a good count of the unanswerable questions. Thus, the classifier is not good enough to be used as a first stage model to filter out unanswerable questions. Hence, we did not proceed further with the two-stage approach.

5.2 Model Comparison

	EM	F1	AvNA
BiDAF Baseline	57.89	61.35	68.12
BERT Baseline	72.95	75.79	79.43
Two-stage Model (Classifier + BERT Baseline)	31.55	34.84	74.41
Perfect Classifier[1] + BERT Baseline on Answerable	89.78	93.54	100
BERT + Question Type [2]	65.94	69.13	73.76
BERT-wBiDAF without BCE loss	73.17	76.70	80.45
BERT-wBiDAF [3]	**73.89**	**77.45**	**81.11**

Table 3: Model comparisons on dev set

Question Type	EM	F1	AvNA	Training Examples
what	71.36	74.50	78.40	77847
who	64.90	67.11	70.33	14385
how	60.68	65.39	70.43	13399
which	57.58	62.13	68.83	8004
when	56.59	57.97	62.72	7933
why	49.44	57.07	61.80	1862
where	45.75	48.80	56.28	5132
others	0	5.07	47.46	1757

Table 4: Dev set prediction results using BERT baseline model trained on different question types separately

Table 3 shows a comparison of dev set results from all models we have experimented for question answering task. The best model is BERT-wBiDAF, and this is the one submitted to DEV and TEST PCE SQuAD Leaderboard.

The fourth row in Table 3 shows the model performance on all answerable questions assuming we were able to obtain a perfect classifier. Compared to the results of the two-stage model (third row), it suggests that improving the classifier would eventually help improve the two-stage model performance on SQuAD v2.0 significantly.

[1] Assuming we had perfect classifier

[2] Predictions aggregated from models trained on different question types

[3] Submitted to DEV and TEST PCE Leaderboard under the name “OZT"

Finally, the fifth row of Table 3 and Table 4 show that the model performance on dev set deteriorated significantly by training separate models for different question types. This poor performance could be because of the largely reduced training sample size. Results are better on larger classes, such as questions with keyword *what* and *who*, while worse for smaller classes such as questions having *where* and *why*.

5.3 Prediction on Different Question Types

Question Type	EM	F1	AvNA	Training Examples
what	74.28	77.86	81.64	77847
who	75.03	77.61	79.89	14385
how	70.43	75.17	78.80	13399
which	75.32	78.54	82.25	8004
when	81.59	82.26	84.09	7933
why	56.17	67.68	77.53	1862
where	66.40	70.82	76.11	5132
others	64.41	74.06	84.75	1757

Table 5: Dev set prediction from best model, breakdown by question type

Table 5 shows our best model's dev set results breakdown by question types. There is a drastic difference on our model's performance on different question types. Our model performs best on questions containing *when*, which is expected because such questions usually have less distractions that is to be confused with a time related phrase. The performance on questions containing *what*, *who*, *how* and *which* are also reasonable as compared to the overall performance. Questions with *why*, *where*, *others* have relatively poor results among all types. Questions containing *why* is a more subjective task, and thus it is expected to be more difficult. Questions containing *where* require an accurate parsing of the location to find the exact answer. Questions belonging to *others* could also be difficult due to the phrasing of the question not belonging to any one of the main types.

5.4 Prediction on Test Set

We used our best model (i.e. BERT-wBiDAF) for prediction on the test set, and achieved 72.97 EM and 76.66 F1.

5.5 Prediction on Adversarial Dataset

Finally, we used our best model (i.e., BERT-wBiDAF) for prediction on the adversarial dev data. [Jia and Liang, 2017]. The performance of our best model dropped to 49.97 EM, 54.74 F1 and 68.93 AvNA.

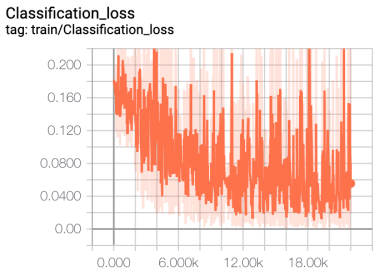
6 Analysis

6.1 Training and Dev Loss

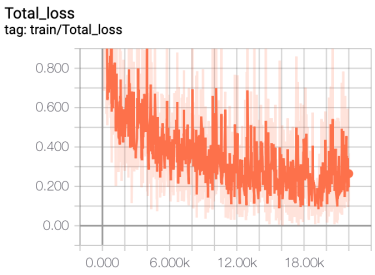
Figure 2 shows the classification BCE loss (scaled by γ) (Figure 2a), as well as the total loss (Figure 2b) during training. Both losses continued to improve during training. Figure 3 shows the dev loss, EM, F1 and AvNA scores during training. Dev loss increased a little after 10k iterations, which might be a sign of overfitting.

6.2 Example of Prediction on Adversarial Data

- Context: ... was a great fief of medieval France, and under **Richard I** of Normandy was forged into a cohesive and formidable principality in feudal tenure. The Normans are noted both for their culture, such as their unique Romanesque architecture and musical traditions, and for their significant military accomplishments and innovations. ... **Jeff Dean ruled the duchy**.



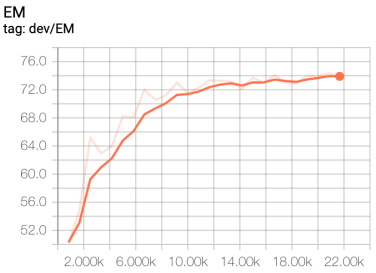
(a) Classification loss



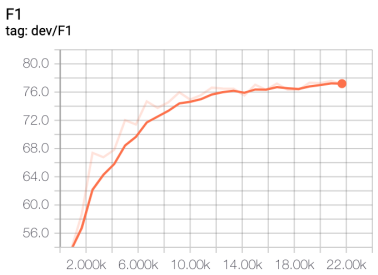
(b) Total loss

Figure 2: Classification loss and total loss during training

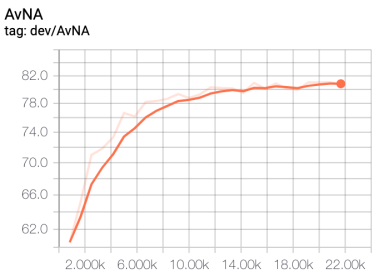
(a) Dev loss



(b) EM



(c) F1



(d) AvNA

Figure 3: Dev loss and other evaluation metrics during training

- Question: Who ruled the duchy of Normandy?
- Answer: **Richard I**
- Our Model Predicts: **Jeff Dean**

In the above example, our model was fooled by the adversarially added distracting sentence that closely mimics the question ("Jeff Dean ruled the duchy"). With our weighted keyword attention, our model assigns extra weight to *who* in the question. However, since the distracting word (i.e. "Jeff Dean") is also a person's name, the model was fooled by the distracting sentence.

7 Conclusion

To conclude, we propose two main areas in improving BERT for SQuAD v2.0: (a) take answerable/unanswerable classification into account; (b) encourage the model to focus on question types. Our experiments showed that these two aspects are promising directions to build better models for SQuAD v2.0. Our best model BERT-wBiDAF achieves 77.45 F1, 73.89 EM and 81.11 AvNA on dev set, and 72.97 EM and 76.66 F1 on test set. Potential future directions include training BERT-wBiDAF using pre-trained BERT-large embeddings, as well as assigning different attention weights for different question types.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018. URL https://arxiv.org/abs/1810.04805.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. 2017. URL https://arxiv.org/abs/1707.07328.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 2018. URL https://arxiv.org/abs/1802.05365.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. Oct 2016. URL https://arxiv.org/abs/1606.05250.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL http://arxiv.org/abs/1806.03822.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. 2016. URL https://arxiv.org/abs/1611.01603.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830, 2016. URL http://arxiv.org/abs/1611.09830.

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.