SQuAD: Eliminate What You Don't Know

Todd Macdonald
Dept. of Computer Science
Stanford University
Stanford, CA 94305
tmacd@stanford.edu

Ashwin Sreenivas
Dept. of Computer Science
Stanford University
Stanford, CA 94305
ashwinsr@stanford.edu

Jessica Zhao
Dept. of Computer Science
Stanford University
Stanford, CA 94305
jesszhao@stanford.edu

Abstract

Machine reading comprehension is an active field of research that involves reading, understanding, and answering questions about a passage of text. The recent publication of BERT, Bidirectional Encoder Representations from Transformers (Devlin et al. 2018), brought significant, near instantaneous, performance gains to this field. Specifically, BERT improved state of the art performance in both SQuAD 1.1 and SQuAD 2.0 (Rajpurkar et al. 2018) simply through the use of BERT's contextual encodings to identify answer spans in a passage. However, our experiments show that models based on BERT still perform poorly at identifying unanswerable questions in SQuAD 2.0. To improve upon this, thereby improving BERT's performance on the dataset as a whole, we ensemble the fully fine-tuned BERT model with an analogous BERT verifier model that shares the same architecture. We show that the verifier's ability to identify unanswerable questions materially improves upon the performance of a BERT model that has been fully fine-tuned on the SQuAD 2.0 dataset.

1 Introduction

The introduction of SQuAD 2.0 significantly increased the difficulty of the popular question-answering benchmark. It's precursor, SQuAD 1.1, simply required that an NLP model identify spans of text from a given context passage to answer a series of questions. Importantly, every question was guaranteed to have an answer span in its corresponding passage. Models such as BERT quickly achieved near human-performance on this challenge. The introduction of SQuAD 2.0 changed this. Primarily, SQuAD 2.0 introduced unanswerable questions; now, rather than simply identifying answer spans, an NLP model would also have to identify questions that had no corresponding answer in the passage. Interestingly, this increase in complexity significantly hurt the performance of previously state-of-the-art models like BERT that achieved near human performance.

In this paper, we extend the work of Devlin et. al (BERT) and Sun et. al (U-Net) to build more powerful question-answering models on SQuAD 2.0. Our approaches for modifying these architectures are detailed in Section 3.

2 Related Work

Our first baseline is a simplified version of the model architecture described in Min Joon et al.'s study Bi-Directional Attention Flow For Machine Comprehension" [6]. Most notable in Min Joon et al.'s work, as noted in the study's title, is the attention layer, which includes both context-to-question and question-to-context attention. The input to the attention layer is the contextual word embeddings of the question and context, which are the hidden states of a bidirectional LSTM, which takes itself as input the word embeddings of both sequences [6].

We used Devlin et al.'s language representation model called Bidirectional Encoder Representations from Transformers, or BERT, in our second baseline. It uses a novel application of Transformers called a multi-layer bidirectional Transformer. Compared to traditional encoder representation approaches, such as with a bidirectional LSTM, that condition separately on the left and right contexts of a sequence, the BERT representations are advantageous in that they are conditioned on both the left and right context concurrently. In figure below, we can see how the input to each transformer unit in a given layer is all of the outputs from the previous layer. For the first transformer layer, the input is all of the sub-word embeddings in the sequence (e.g., the question and passage, as in SQuAD).

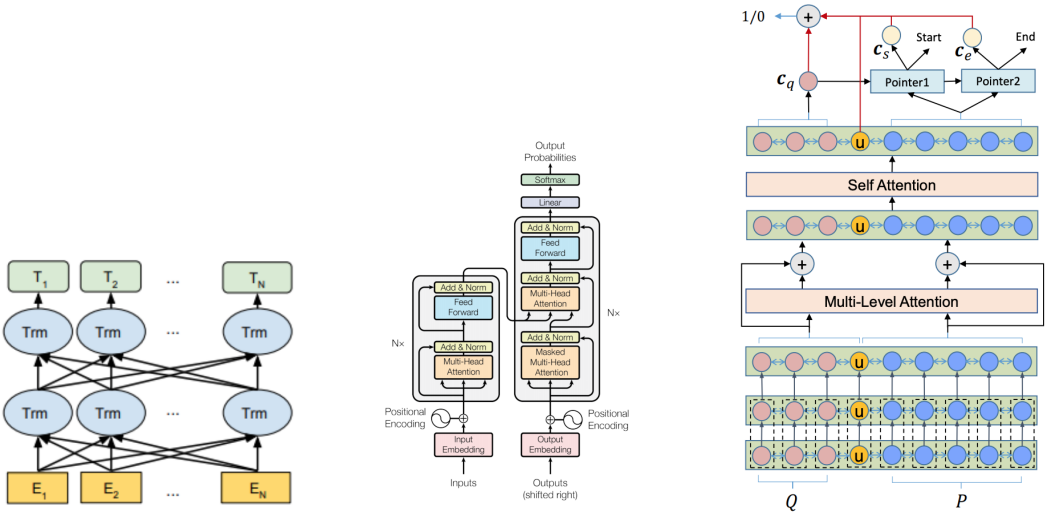

Figure 1: Baseline architectures. (a) Devlin et al.'s BERT [1]; (b) Vaswani et al.'s Transformer [8]; (c) Sun et al.'s U-Net [7].

One of the building blocks to the BERT baseline, as well as multiple of our experimental models, is an attention mechanism known as the Transformer, introduced by Vaswani et al. [8]. Unlike with RNN's, a Transformer does not use an autoregressive structure, in which each token of the input sentence would need to be sequentially passed in as input. As a result, the Transformer can process all of the tokens in the input or output in parallel. These computational blocks consist of multi-headed self attention, residual connections, and a feed forward layer. Self-attention involves attending an input with a trainable representation, and multi-headed attention involves using several copies of a simpler attention mechanism, such as dot-product attention, whose representations are then concatenated as input into a feed-forward layer.

We also use Sun et al.'s machine comprehension model called U-Net for our third baseline [7]. U-Net seeks to address the unanswerable questions of SQuAD 2.0 by introducing an answer pointer to predict answer spans, a no-answer pointer to avoid selecting any spans in the case of unanswerability, and an answer verifier to determine the unanswerability probability of a question. The authors use a combination of non-PCE and PCE embeddings, including GloVe and ELMo. These three subtasks are unified in an end-to-end pipeline whose implementation we further describe and modify in Section 3.5.

3 Approach

3.1 Baseline: BiDAF

This is the baseline that was part of the starter code for the default final project. Since we did not modify this code at all, and since it is not central to our paper, we omit describing it here. However, we will compare its performance to the rest of our models in Section 4.

3.2 Baseline: BERT

For our baseline, we use a PyTorch pre-trained BERT model, from Huggingface, that is adapted for SQuAD. For predictions, we use the additional linear layer, on top of the vanilla BERT model, that outputs logits for the start and end indices of the predicted answer [3].

To identify unanswerable questions, an OOV token is appended to the beginning of the input to the BERT model. The log probability of a question being unanswerable is equal to $p_{start}(0) + p_{end}(0)$, where $p_{start}(0)$ and $p_{end}(0)$ are the logits for the start and end indices of the answer being the OOV token.

For best model performance, we follow the pre-training and fine-tuning procedures as mentioned in Devlin et al. In pre-training, the model is given an input sequence in which some of the words are masked. The prediction task is then to predict the masked words using the context of the other words, which Devlin et al. refer to as the "masked language model" objective [1]. The pre-training corpus consists of over 3 billion words from BooksCorpus and English Wikipedia, many orders of magnitude larger than the SQuAD dataset, which has only about 150,000 questions, and as such enables significant transfer learning [1].

Since the Huggingface implementation of BERT includes saved model weights from the pre-training step, we use these weights instead of pre-training the model ourselves. For the fine-tuning step, we then use the BERT model, initialized with weights from the pre-training step, with an additional linear layer, and train this model in SQuAD. From the output of the final (12th) transformer layer, we get the contextual embeddings for each sub-word in the input sequence. The linear layer then converts these contextual embeddings into two logits that represent the probability that the start and end of the answer sequence are that sub-word.

To train this model, we average the multi-class cross-entropy loss over the start and end indices of the model's output. The results of this approach are described in Section 4.

3.3 Baseline: U-Net

For our last baseline, we use Sun et al.'s U-Net Machine Comprehension model, from FudanNLP, which introduces an answer verifier to address the non-answerable questions in SQuAD 2.0 [2]. U-Net takes in GloVe and ELMo embeddings as input and feeds them through several layers of Multi-Attention and Self-Attention before arriving at a three-part loss function focused on capturing answerability. The results of this approach are described in Section 4, and we detail how we build upon this model in Section 3.5

3.4 BERT + Answer Verifier

This approach is the ensemble of two other models. First, we use the Baseline BERT model from section 3.1 that has been fine-tuned on the SQuAD 2.0 dataset and has an additional linear layer for the start and end indices of the predicted answer.

Second, we independently train another BERT model that has the same architecture as the baseline BERT model. However, this model only has one output in its final fully connected layer: whether or not the question is answerable. We adapt the SQuAD 2.0 dataset to have the "answers" to the questions+context pairing to be whether or not the question is answerable. Then, we can train this model using a simple binary cross-entropy loss.

Finally, the two models above are ensembled. The second model is used first to predict whether or not a question is answerable. If it isn't, the entire model simply predicts that the question is not answerable. If it is, the first model is run and the output of the first model is used as the final answer. (Note, in practice, both models are run simultaneously and the results of the first model are only used if the second model predicts that the question is answerable). The results of this approach are described in Section 4.

3.5 BERT + U-Net

In this approach, we build upon the U-Net Machine Comprehension architecture proposed by Sun et al. Instead of GloVe and ELMo embeddings as model input, we use embeddings from the Baseline

BERT uncased model from Section 3.2 that has been fine-tuned on SQuAD 2.0. We considered several ways to incorporate these embeddings and built three custom models before deciding to focus on the third. In the first model, we aimed to replace GloVe with BERT embeddings at the preprocessing step. However, because the BERT embeddings are contextual while the GloVe embeddings are not, we could not appropriately construct a token-to-vector mapping without losing the contextual advantages of BERT. In the second model, we attempted to keep GloVe embeddings and replace ELMo embeddings with BERT. However, we faced challenges as both the GloVe and ELMo embeddings were based on the allennlp tokenizer while BERT uses its custom WordPiece tokenizer, which for a single example, generally produced more tokens than the former. This difference resulted in dimension incompatibilities. Therefore, in our final model, we decided to focus solely on passing BERT embeddings through the U-Net model to reduce dimensional inconsistencies.

To achieve a rich-text representation, we use both case-sensitive and case-insensitive information by extracting the original paper's part-of-speech and name entity recognition features with case-sensitive input tokens, concatenating original, lowercased and lemmatized text representations, and finally incorporating our additional BERT contextual data with case-insensitive input tokens.

Then, following the authors' approach, we build an input layer composed of a fused representation of question, passage, and universal node focused on learning answerability. The collective representation allows for an end-to-end prediction model with a consolidated loss function rather than a fragmented pipeline that isolates the answer verifier segment from the rest of the model. To arrive at this fused representation:

1. Each word in question and passage is a $d-$dimensional embedding, where $d = 768$ following the shape of BERT embeddings. We concatenate the question, U-node, and passage representations into a single vector $V \in \Re^{dx(mxn+1)}$
2. We use a BiLSTM of two layers to produce low-level and high-level semantic representations H_l and H_h.
3. We concatenate H_l and H_h and run it through a BiLSTM of one layer to produce the full representation H_f.
4. We perform multi-level attention between questions and passages to find $\hat{H}_l, \hat{H}_h, \hat{H}_f$.
5. We perform self-attention. To do so, we use a BiLSTM to concatenate H and $\hat{H}$.

   $$H^A = BiLSTM([H^l; H^h; H^f; \hat{H}_l; \hat{H}_h; \hat{H}_f])$$

   Then we concatenate the original V:

   $$A = [V; H^A]$$

   Then we apply self-attention upon the fused information A to arrive at the attended $\hat{A}$.
6. To arrive at the final fused representation,

   $$O = BiLSTM[H^A; \hat{A}]$$

We use the loss functions of three prediction tasks:

1. Answer pointer, to detect answerability and predicts an answer from the passage text spanning (a, b):

   $$\mathcal{L}_A = -(log\alpha_a + log\beta_b)$$
2. No-answer pointer to detect non-answerability, where α_0 and β_0 correspond to the position of the universal node:

   $$\mathcal{L}_{NA} = -(log\alpha_0 + log\beta_0)$$
3. Answer verifier, which uses a prediction p^c built upon information from the question, passage, and universal-node. $\delta \in \{0, 1\}$ indicates answerability.

   $$\mathcal{L}_{AV} = (\delta log p^c + (1 - \delta)(log(1 - p^c)))$$

To train, we combine the losses into a single function:

$$\mathcal{L} = \delta\mathcal{L}_A + (1 - \delta)\mathcal{L}_{NA} + \mathcal{L}_A V$$

Finally, we re-add the linear layer to arrive at the start and end logits.

3.6 Code and Original Approach

Our three baseline models are each constructed from their repositories cited below.

Our BERT + Answer Verifier model is adapted from the Huggingface BERT repository implemented in PyTorch. We extended this model to include our additional verifier model; this involved modification of the actual final prediction layer, and the costs associated with it for training (detailed in section 3.4). We also wrote the code to combine the predictions of BERT with the verifier.

Our BERT + U-Net model is adapted from the FudanNLP repository but incorporates several new bodies of code. For each of our three experimental models under the U-Net exploration, we constructed independent training and model classes as well as many helper functions to modify BERT examples to feed into the U-Net architecture. Our approach is original in its combination of case-sensitive and case-insensitive data, the former of which is valuable in capturing named entities that often form answers and the latter of which is valuable in generalizing textual content to perform more robustly on unseen data.

Additionally, we wrote two pipelines to translate BERT and U-Net predictions to the desired csv format in order to submit to the leaderboard.

We also wrote code to perform quantitative and qualitative error analysis to illuminate patterns in the performance of question-passage pairs and categorize scores by answer type, which we discuss in Section 5.

4 Experiments

4.1 Data

We are using the modified CS224N SQuAD 2.0 datasets, which are a subset of the original SQuAD 2.0 datasets released by Rajpurkar et. al [5].

4.2 Evaluation Method

For our metrics of choice, we intend to follow the rest of the SQuAD 2.0 leaderboard and optimize the F1 and EM scores obtained by all of our models.

Note that we have three baselines here. The first baseline is the one from the default starter code. The second baseline is the Huggingface BERT model that has been finetuned on the SQuAD 2.0 dataset (described in Section 3.1). The third baseline is U-Net (described in Section 3.2).

We have built upon these baselines using two other approaches. The first is the BERT+Answer Verifier model, described in Section 3.3, that should be compared to the BERT baseline. The second is the BERT+U-Net model, described in Section 3.4, that should be compared to the U-Net baseline.

4.3 Experimental Details

The Huggingace BERT baseline implementation was fine-tuned on the SQuAD 2.0 dataset for 6 hours for two epochs on an Azure NV12, using two GPUs. Since computational constraints now allowed for it, we fine-tuned the entire model with a low learning rate, instead of restricting ourselves to the top layers alone. We used the following hyperparamters to fine-tune:

- Batch size: 12
- Learning rate: 3×10^{-5}
- Max sequence length: 384
- Doc stride: 128
- Epochs: 2

Similarly, we tuned the two ensembled components of the BERT+Answer verifier layer on the SQuAD 2.0 dataset on an Azure NV12, using two GPUs. We also fine-tuned the entirety of both models using the same hyperparameters as above.

As for U-Net, we tuned the FudanNLP U-Net implementation on the SQuAD 2.0 dataset for 6 hours over 30 epochs on Azure NV12. We used the following parameters to fine-tune:

- Batch size: 32
- Learning rate: 2×10^{-3}
- Max sequence length: 384
- Epochs: 15 and 30 (separate runs)

We tuned the BERT + U-Net model, adapted from the FudanNLP repository, on the SQuAD 2.0 dataset for 12 hours over 6 epochs using Azure NV24. We used the following parameters to fine-tune:

- Batch size: 32
- Learning rate: 2×10^{-3}
- Max sequence length: 384
- Doc stride: 128
- Epochs: 6

4.4 Results

Running the experiments described in Section 3 yielded the following results on the SQuAD 2.0 dev and test sets:

Model	EM	F1	EM-HA	EM-NA	F1-HA	F1-NA
Baselines						
BiDAF	57.63	60.86	60.58	52.61	67.18	52.61
BERT + Linear Layer	65.76	69.22	74.30	57.92	81.52	57.92
U-Net (15 epochs)	46.38	51.28	52.61	40.66	62.85	40.66
U-Net (30 epochs)	66.70	70.95	55.32	77.15	64.30	77.15
Ensembles						
BERT + U-Net (Dev)	59.56	64.10	57.22	61.71	66.71	61.71
BERT + Verifier (Dev)	67.03	70.37	71.24	63.16	78.21	63.16
BERT + Verifier (Test)	67.46	71.15	–	–	–	–
Other						
224N Official Baseline (Test)	56.29	59.92	–	–	–	–
Human Performance	86.83	89.45	–	–	–	–

Figure 2: Baseline and Experimental Results. Note that all results should be compared to the PCE leaderboard.

5 Analysis

5.1 Qualitative Evaluation

5.1.1 BERT + Answer Verifier

To better understand how our top models, BERT and BERT + Answer Verifier, do on different types of answers, we divide the results into subsets dependent on certain properties of the predicted answer strings.

Answer attribute	BERT EM	BERT F1	BERT+Verifier EM	BERT+Verifier F1
Single word answer (nonempty)	59.87	61.88	61.36	63.42
Two word answer	57.17	60.57	58.39	61.90
> 1 word answer	51.11	58.13	52.74	60.05
Answer is digit	62.75	63.92	65.65	66.93
Repeated > 3 times (non-empty)	48.57	50.42	48.31	50.02
Repeated > 3 times & multi-word	43.42	47.39	41.67	44.80
Predicted answerable	54.17	59.44	55.77	61.23
Predicted unanswerable	87.93	87.93	84.75	84.75

Figure 3: EM and F1 scores for the BERT and BERT + Answer Verifier models on different subsets of the dev training set.

In both models, we see that the EM score is much higher in predicted answers that are a single word while the F1 scores are relatively similar. For instance, the F1 score with the BERT model is 61.88

and 58.13 with one word and greater than one word answers, whereas the respective EM scores are 59.87 and 51.11. This result makes intuitive sense, as shorter answers are often named entities or numbers, for which it is easier to identify the exact beginning and ending indices. Digits such as dates, a subset of single word answers, have even higher EM and F1 scores, at 62.75 and 63.92 for the vanilla BERT model.

Interestingly, we see that both the BERT and BERT+Verifier models perform significantly worse when predicting several questions to have the same answer. For instance, the vanilla BERT model achieves a 58.13 F1 on multi-word answers but only a 47.39 F1 score on multi-word answers that the model predicts at least 3 times (for 3 separate questions). This result indicates that the models tend to over-predict common answers to questions.

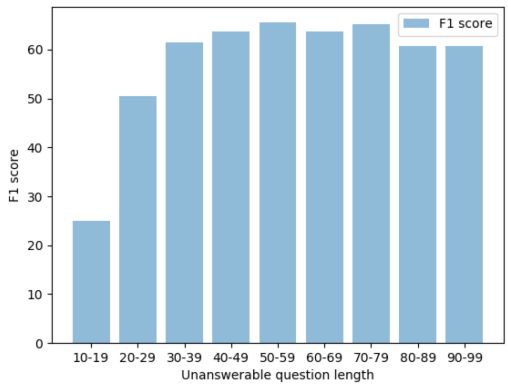

Figure 4: Comparison of question length to F1 score for unanswerable questions. F1 scores are for the BERT + Answer Verifier model.

Overall, the question length of unanswerable questions does not have a strong correlation with model performance for questions at least 30 characters long. While the result for the 10-19 character question length can be discounted, as there are only 4 such questions, the 20-29 character length interval has 111 such questions, which is more significant and may indicate that it is more difficult to predict unanswerable questions if the question is quite short, possibly due to a lack of context in the question.

5.1.2 BERT + U-Net

We observe that BERT + U-Net is outperformed by both the U-Net baseline and BERT + Answer Verifier. This may be for a few possible reasons. First, we trained BERT + U-Net for 6 epochs over 12 hours on NV24, compared to U-Net for the full 30 epochs over 6 hours on NV12. Therefore, the resulting numbers may be partially attributed to the lower number of epochs. Notably, our 6-epoch model outperforms vanilla U-Net's 15-epoch model on every measure, suggesting the robustness of BERT embeddings and a higher score if trained over the full 30 epochs.

Since BERT embeddings are already fine-tuned, it is possible that passing them through U-Net's multiple layers of attention, on top of BERT's existing layers of attention, dilutes the contextual information already present in the vectors. This suggests some redundancy in U-Net's attention layers, resulting in diminished performance compared to BERT + Answer Verifier. An improved BERT + U-Net architecture might focus on the unanswerability pointer of U-Net to improve the score gap between answerable and unanswerable questions, and place less focus on learning answer spans, a task on which BERT with simple modifications already performs well.

5.2 Quantitative Analysis

5.2.1 BERT + Answer Verifier

In comparison to the BiDAF baseline model, we see that the baseline BERT, using only a linear layer on top of the BERT model, has significantly better EM and F1 scores for every category in the table.

As we can see, the fine-tuned BERT model takes significant performance penalties because it is unable to effectively identify when a question is unanswerable. While it enjoys an F1 score of 81.52 on questions that have an answer, it drops to 57.92 on questions that do not have an answer.

Our BERT + Verifier ensemble is designed to better this discrepancy. By preemptively identifying questions that are not answerable, it is able to significantly boost performance in unanswerable questions. Notice that this approach boosts F1 scores of unanswerable questions to 63.16, from the 57.92 for baseline BERT. This significant performance gain comes only at the cost of a slight penalty to the F1 score of answerable questions. The EM metric also receives benefits analogous to the F1 score.

Our model also does not overfit to the train and dev sets as can be seen by the results in Figure 2 - both the EM and F1 scores on the test set are in line with those on the dev set.

5.2.2 BERT + U-Net

Out of all the models, BERT + U-Net achieves the smallest differential of approximately 5 points between F1-Has Answer and F1-No Answer scores, suggesting that its subtask pointers for answerability are working to balance out predictions for both categories of answers. The next closest differential is approximately 15 points for BERT + Verifier. Additionally, we observe a nearly 4-point improvement in F1-No Answer (61.71, compared to the BERT baseline of 57.92), suggesting the model's strength in detecting unanswerability may increase with more training time.

6 Conclusion

The introduction of unanswerable questions in SQuAD 2.0 significantly hurt the performance of models that performed at near human-levels. In this paper we have shown that state-of-the-art models such as BERT suffered this performance penalty because they were substantially worse at identifying unanswerable questions than they were at answering questions that did indeed have answers. To combat this, we introduced an ensemble model, the BERT+Verifier, which consisted of two models with similar architectures: the first was a BERT model that was trained to identify answer spans; the second was a model based off of BERT whose sole purpose was to identify unanswerable questions. We show that this ensemble is able to boost the overall performance on SQuAD 2.0 since the verifier model is able to adequately screen out unanswerable questions.

Our work also highlights the large computational resources required of model architectures such as BERT and possible future directions of research that could address this issue. While BERT utilizes a model architecture that is highly parallelizable, this parallelization requires immense computational resources. Running BERT on Azure's NV-12 instance, we could only use a batch size of 4, and as a result each epoch would take about 3 hours, compared to 15 minutes for the vanilla U-Net. Therefore, while BERT-related models perform better over an equal number of epochs, they require more computational resources to do so. Furthermore, the pre-training step of BERT is even more computationally costly. Google researchers pre-trained the large BERT model on 16 Cloud TPU's over 4 days; at the current price of $\$4.5$ per TPU per hour, the computational cost is around $\$7000$, above the budget for many researchers [1]. Therefore, while BERT has proven to be a successful model architecture, future directions of research could explore using this new model architecture in a computationally constrained environment.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] FudanNLP. The official implementation of paper u-net: Machine reading comprehension with unanswerable questions. `https://github.com/FudanNLP/UNet`, 2018.

[3] huggingface. The big extending repository of transformers: Pytorch pretrained models for google's bert, openai gpt gpt-2, google/cmu transformer-xl. `https://github.com/huggingface/pytorch-pretrained-BERT`, 2019.

[4] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[5] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[6] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

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

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.