LHAMa SQuAD Goals: Adversarial Question Generation and Model Exploration for Question Answering

Allen Kim (6812605923), Huy Nghiem (4128492129), Malika Seth (5458237216), Yi Wang (1114413388)
Viterbi School of Engineering
University of Southern California
Los Angeles, CA 90089
{allenkim, knghiem, malikase, wang796}@usc.edu

Abstract

Question answering is a well researched topic in the field of Natural Language Processing (NLP) that focuses on building systems to answer questions. Although much has been researched in the field of question answering, the extent to which these systems are able to comprehend and generate questions is still uncertain. The current state of question answering systems are able to generate correct and relevant answers given answerable questions, but when presented with adversarial questions, these systems are often unable to discern whether the answer exists in the text. Our project focuses on trying to understand the limitations of question answering models through the creation of adversarial unanswerable questions. We propose two methods in which we generate true adversarial unanswerable questions and use these questions along with the Stanford Question Answering Dataset (SQuAD) 2.0 to test against the capabilities of Bidirectional Encoder Representations from Transformers (BERT) and other architectures built on top of BERT. While our unanswerable question generation methods proved to be novel and easy to use, our additional architectures for the question answering task had mixed results.

The code and models we have used is all open source and can be found online at `https://github.com/USC-LHAMa/CSCI544_Project`.

1 Introduction

Adversarial questions are one of the many challenges posed in NLP. Given an adversarial question, a question answering model will likely have a difficult time discerning whether the answer is located in the given text or if the question is truly unanswerable. In many situations, these models will generate plausible answers that do not necessarily contain the correct answer. Thus, it is important to understand the limitations of certain question answering models and how they perform when given adversarial questions. These adversarial questions must be relevant in the corresponding paragraphs to serve as an effective detractor. In other words, a good unanswerable question should "look" like it has a correct answer from the paragraphs, but not so in reality.

There are various different methods for training and evaluating language representations. We chose to use BERT, which is a method of pre-training representations [1]. We introduce and explore an enhancement to the SQuAD dataset against different BERT models that is aimed to increase the accuracy and F1 scores of the base model. In this paper, we will discuss our efforts to generate these adversarial questions and test these generated questions with SQuAD 2.0 against custom BERT models. We hypothesized that with the addition of newly generated adversarial questions to the SQuAD 2.0 dataset, the F1 scores for all BERT model variations would increase. In order to evaluate this hypothesis, we generated adversarial questions deemed to be unanswerable using various methods which will be discussed later in this paper. These custom questions were appended to the SQuAD 2.0 dataset and we ran this augmented dataset against four different types of BERT models (vanilla BERT, LSTM, CNN, multi-linear). Our main contributions are the definition of two new methods of generating unanswerable, adversarial questions for use in the SQuAD question answering NLP task, nearly 200 examples of such unanswerable questions appended to the SQuAD 2.0 training set (which we have dubbed SQuAD v2.1), and the exploration and experimentation of additional architectures on top of BERT for the SQuAD question answering task. The code and models we have used is all open source and can be found online at `https:`

//github.com/USC-LHAMa/CSCI544_Project.

2 Related Work

2.1 SQuAD 2.0 and Adversarial Question Generation

SQuAD is a dataset that consists of various crowd-sourced questions based on a set of Wikipedia articles. For each question, the answer is based on a segment of text from the corresponding reading passage. Each question can either have a definite answer or the question can be unanswerable. SQuAD 1.1 contains over 100,000 question-answer pairs on over 500 articles. SQuAD 2.0 consists of the original 100,000+ question-answer pairs from the original SQuAD 1.1 dataset and includes over 50,000 unanswerable adversarial questions. In the SQuAD 2.0 paper ”Know What You Don’t Know” [2], the problem of unanswerable questions is discussed. To briefly summarize, questions where a correct answer is not guaranteed to exist pose as affective distractors towards language models.

Our research is partially inspired by the the paper ”Adversarial Examples For Evaluating Reading Comprehension Systems” [3]. This paper focuses on testing whether certain language modeling systems can correctly answer questions about paragraphs that have adversarially inserted sentences with the SQuAD dataset. The inserted adversarial sentences are used in order to target the model’s instability. In other words, these sentences are to confuse the model in answering incorrectly by containing words that are common with the given question but are not intended to change the actual answer to the question. The paper found that the accuracy of 16 different published models dropped from an average of a 75% F1 score to a 36% F1 score. Thus, current models do poorly given adversarial sentences and are only stable with straightforward passages.

We use these papers to build off of existing research to try to understand the limitations of current language understanding models given adversarial questions.

2.2 BERT

BERT is a method of pre-training language representations, meaning that a general-purpose ”language understanding” model is trained on a large text corpus like Wikipedia that can be used in downstream NLP tasks such as question answering. BERT uses Transformers [4] and is bidirectional, meaning it processes text from left-to-right and right-to-left for more context. To create the language representation, BERT pre-trains on two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP), both self-supervised tasks. For MLM, a percentage of words in a sentence are masked and the model predicts the hidden words. In NSP, the model predicts whether one sentence follows another.

The pre-training of BERT is a computationally-expensive and time-consuming task. To leverage it for downstream NLP tasks, however, a process called fine-tuning is used. In fine-tuning, an additional layer(s) such as a fully-connected linear layer with randomly initialized weights is added to the end of the base-BERT network with its pre-trained weights and then trained for the new task. All of the weights in the entire network will change, but the weights for the new layers will be affected the most and the fine-tuning process is much shorter than the initial pre-training.

Google open sourced and released the code for BERT along with its paper, written in TensorFlow. HuggingFace created a PyTorch implementation of BERT along with other Transformer architectures in a popular and easy-to-use package [5]. For our project, we leverage the HuggingFace version and extend it to meet our needs.

3 Methodology

To attempt performance improvements on the SQuAD 2.0 dataset, we take the following approach: Define new methods to create adversarial questions Leverage pre-trained BERT models to perform the SQuAD task provided by HuggingFace [5]

To baseline our enhancements, we fine-tuned the standard model for the SQuAD task provided by HuggingFace, which consists of a single linear layer on top of the BERT pre-trained model that outputs predictions for start and end points for the answer. We also fine-tuned the HuggingFace model against our enhanced v2.1 SQuAD dataset for comparison as well. All of our experiments used the same hyperparameters when training, and we evaluated against the SQuAD 2.0 dev set, acting as the standard test set for all of our models. Specific hyperparameter values can be found in the appendix.

3.1 Creating Adversarial Questions

There have been many attempts to increase the capabilities of Question-Answering models by training them on adversarial questions, defined as queries written to intentionally trick the model to select the wrong answers. Typically, the questions are created by incorporating and manipulating data from the original passages to present erroneous information to them as legitimate queries. Jia and Lang [3] presented variants of their AddSent model, which appends tangentially related but misleading information to the passages, and queries the model with adversarial questions based on these imprecise addendum. Wallace and Boyd-Garber [6] proposed another method to create adversarial questions on trivial challenges by entirely human interactions. In most cases, researchers report an increase in performance and robustness after training their models on additional adversarial sets [7].

In our work, we attempt an original approach to generate adversarial questions. Based on the understanding that the attention-based models answer queries by focusing on certain parts of the passage, we create adversarial questions by combining information in the same passage and generate new queries based on this new, but erroneous information, e.g. switching named entities or other words/phrases. We argue that this approach is effective at generating legitimate adversarial questions due to two reasons. First, since the adversarial questions were created entirely based on local information (the passage itself), instead of outside sources, they should appear very similar in terms of legitimacy in comparison to the original questions. Second, this approach is low-cost since no additional data is needed in the process. We successfully created 178 original adversarial questions to be appended to the original SQUAD 2.0 dataset, christened as SQUAD 2.1.

We first identify a sentence S1 of interest. Then we select a sentence S2 in the same passage that describes another object. Some phrases of S2 are appended in ways that make syntactical sense into S1 to generate a new sentence S*, which contains erroneous information due to its hybrid origin. Adversarial questions of the who/what/when forms are created based on S*, typically by targeting some parts of S* as a potential answer. As a result, these questions do not have a legitimate answer. A robust model should indicate a ¡No Answer¿ response for these questions. Otherwise, the model's answers may also yield valuable insights into their internal mechanisms.

For a deeper look into our methods, we reference our raw data for unanswerable question generation: https://bit.ly/2DMIcTk.

3.2 Leverage Pre-Trained BERT Models

Using pre-trained models has become quite ubiquitous and we also decided to go this route. When it was released, Google's BERT framework produced state-of-the-art results on numerous NLP tasks so we were eager to use it. HuggingFace also has a popular, easy-to-use, and open source PyTorch implementation that includes a basic runner of the SQuAD task that we leveraged.

The HuggingFace implementation of the SQuAD task includes a single linear layer on top of pre-trained BERT that can be fine-tuned with a SQuAD-compatible training set. We baselined with this model against both the SQuAD v2.0 training set as well as our augmented SQuAD v2.1 data set. Additionally, we created the following new architectures on top of pre-trained BERT:

- **Multiple linear layers**: Rather than a single linear layer, which distills knowledge from pre-trained BERT to distinct start and end points for the SQuAD task, we thought additional linear layers with ReLU nonlinearities between them could learn more about the patterns in the data.

- **Convolutional layer**: In the paper "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling" [8], the authors describe the efficacy of using convolutional networks for sequence modeling tasks, rivaling and surpassing recurrent networks in some instances. We wished to explore this by implementing a single convolutional layer on top of pre-trained BERT along with a ReLU nonlinearity before the two-value output.

- **LSTM layer**: Text data, since it is sequential in nature, is often manipulated in NLP tasks with recurrent architectures such as RNN and LSTM. We hoped to improve upon the single linear layer by adding an LSTM layer that would exploit this information before the final two-value output.

We performed the same fine-tuning process with BERT + [New Architecture] to obtain results such as F1 score and accuracy on the evaluation dataset, which was the SQuAD 2.0 development set in all instances so that we were able to compare across all models. Final results can be seen in the Results section as well as more detailed information in the Appendix.

4 Results

4.1 Basic F1 Score Comparison

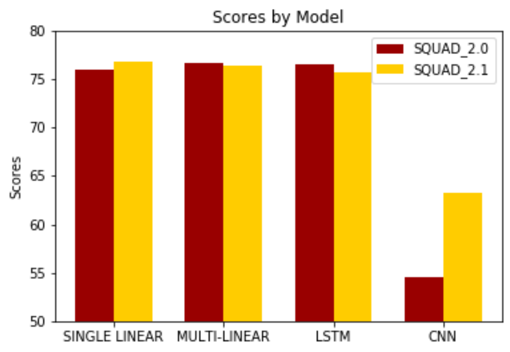


Figure 1: F1 for Models

It is observed that the mult-linear layer (comprising of three layers) on top of BERT outperforms all other models for the original SQUAD 2.0 dataset. While single linear layer over BERT performs best for the SQUAD 2.1 dataset, followed closely by multi-linear layer. Moreover, contrary to our hypothesis, the F1 score over SQUAD 2.1 has only increased for the single layer and CNN models.

However, overall it is noticed that CNN has the worst performance. This is possibly due to the fact that CNN is suitable for spatial data such as images and video processing. Convolutions and pooling operations lose information about the local order of words, such that sequence tagging or entity extraction is harder to fit into a pure CNN architecture. The type of neural network that performs better when dealing with text data, depends on how often the comprehension of global/long-range semantics is required. For tasks like question-answering and translation where length of text is important, we realized it is preferable to go with RNN variants (e.g. LSTM).

4.1.1 F1 Score for Answerable Questions

While analyzing performance on the answerable questions, LSTM was found to be the best model when evaluated against both the SQUAD datasets, with an F1 score of 80.78. This further proves our

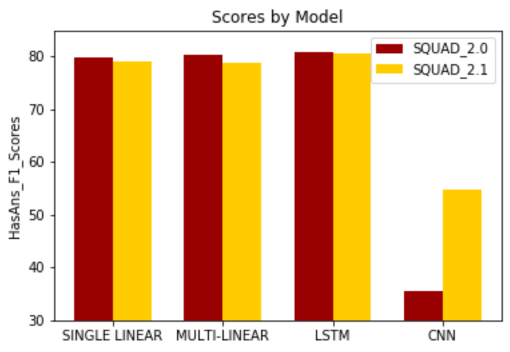


Figure 2: Answerable Question F1 for Models

point regarding RNN variants being highly suitable for text processing.

4.1.2 F1 Score for Unanswerable Questions

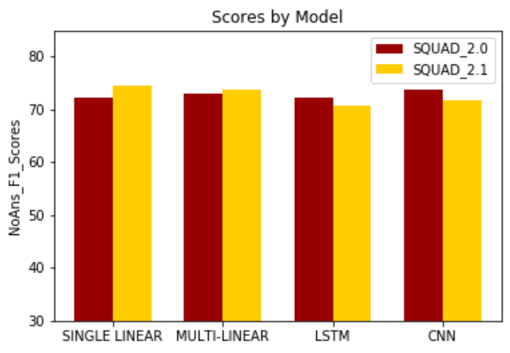


Figure 3: Unanswerable Question F1 for Models

Surprisingly, in the case of unanswerable questions, CNN is noticed to have outperformed all other models. Based on this architecture's poor performance on the set of questions with answers, however, we remain skeptical on whether this indicates the CNN model is actually effective at detecting unanswerable questions. This is an interesting area that can be further explored.

4.2 Qualitative Analysis

In addition to the evaluation metrics presented above, we were also interested in analyzing our models' qualitative performance with respect to adversarial questions. We manually investigated the answers our models produced for these "impossible" questions, as they did not contain any correct answer inside their passages. They, however, could contain plausible answers, which were phrases of the passage that could partially answer the question, but not holistically.

A (appendix) demonstrates some pairs of adversarial questions along with the corresponding answers produced by each of our model. Each question targets a specific passage, which in turn is a part of a topic. The first two questions showcased our models' ability to correctly predicts ¡No Answer¿ across the board. The third question about Steam Engine is an example of when the models diverged in their prediction. Notably, both CNN- and LSTM-incorporated models yielded similar responses as the given plausible answer "Rankine", demonstrating their ability to pick up cues in the context of the passage.

The difference in performance becomes more pronounced in the question on "Computational Complexity", where clearly our CNN yielded a truncated response compared to all other models with respect to the reference plausible answer. Even more so in the last question, CNN model yielded an entirely off-based answer while others did not. These results are mostly consistent with the overall evaluation metrics that CNN-based model did not perform as well as others.

5 Future Work

It is important to note that our set of adversarial questions is very much a work in progress. First, 178 questions is still a very small number with respect to the thousands of questions in the original SQuAD dataset. We recognize that our questions' effectiveness in improving the models' learning capabilities may be hampered by the scale. We plan on generating significantly more questions in the future to more accurately assess our methodology's prowess. We plan on recruiting Amazon ® Mechanical Turk to accomplish this goal. Furthermore, we also desire to explore other question generating mechanisms based on the proposed framework to design questions geared to reveal insights about models' attention mechanism.

Regarding the additional architectures, for our experiments we used the recommended hyperparameters provided in the Google BERT source code repository. Since those hyperparameters were tuned specifically for the single linear layer architecture, they were likely not well-suited for our multi-linear, CNN, and LSTM models. In the future, we should perform hyperparameter tuning specifically for each of the models on the SQuAD dev set, and receive the test F1 from submitting to the SQuAD site with our best parameters.

We also noticed that our F1 scores were well below the state-of-the-art scores that BERT posted on the SQuAD task. Reading further into the documentation, we noticed that those scores were achieved by fine-tuning on the BERT-large pre-trained model. Due to time and computational constraints, we used the BERT-base pre-trained model, which had fewer parameters and thus did not learn as much. Lastly, we could explore other hyperparameters specific to our architectures, such as number of layers and bidirectionality for LSTM and kernel size and stride for CNN.

6 Conclusion

In this paper, we presented two new methods for generating unanswerable, adversarial questions for use in the SQuAD dataset and experimented with three architectures for the SQuAD question-answering task, baselining it against the given single linear layer provided by the HuggingFace implementation.

For generating new unanswerable questions, the methods we created led to the creation of nearly 200 new adversarial questions we used for augmenting the SQuAD v2.0 training set and fine-tuned various models to evaluate against the SQuAD v2.0 dev set. Our methods lead to relatively fast and low-effort question creation, though it is still a manual, human-powered effort.

The additional models of multi-linear, convolutional, and LSTM layers on top of the BERT pre-trained language model led to mixed results. It was observed that CNN with BERT led to a lower overall F1 when implemented over both the datasets. However, the results of F1 score for adversarial questions were strikingly opposite in nature, with CNN surpassing all other models.

Our main contributions were the creation of two methods to create unanswerable, adversarial questions as well extending the default HuggingFace linear model for the question answering task and running experiments against both the SQuAD 2.0 and our SQuAD 2.1 training data, allowing for further diversification of SQuAD and exploration of different models built on top of BERT.

7 Acknowledgement

The authors would like to thank Dr. Peng and Rujun Han for their insight and assistance they provided on this project and for the knowledge they have given us during the entire semester.

References

1 Jacob Devlin, Ming-Wei Chang, Kenton Lee, & Kristina Toutanova, ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv:1810.04805 [cs.CL], Oct. 2018.

2 Pranav Rajpurkar, Robin Jia, & Percy Liang, ”Know What You Don’t Know: Unanswerable Questions for SQuAD,” arXiv:1806.03822 [cs.CL], Jun. 2018.

3 Robin Jia, & Percy Liang, ”Adversarial Examples for Evaluating Reading Comprehension Systems,” arXiv:1707.07328 [cs.CL], Jul. 2017.

4 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, & Illia Polosukhin, ”Attention Is All You Need,” arXiv:1706.03762 [cs.CL], Jun. 2017.

5 Pranav Rajpurkar, Robin Jia, & Percy Liang, ”Know What You Don’t Know: Unanswerable Questions for SQuAD,” arXiv:1806.03822 [cs.CL], Jun. 2018.

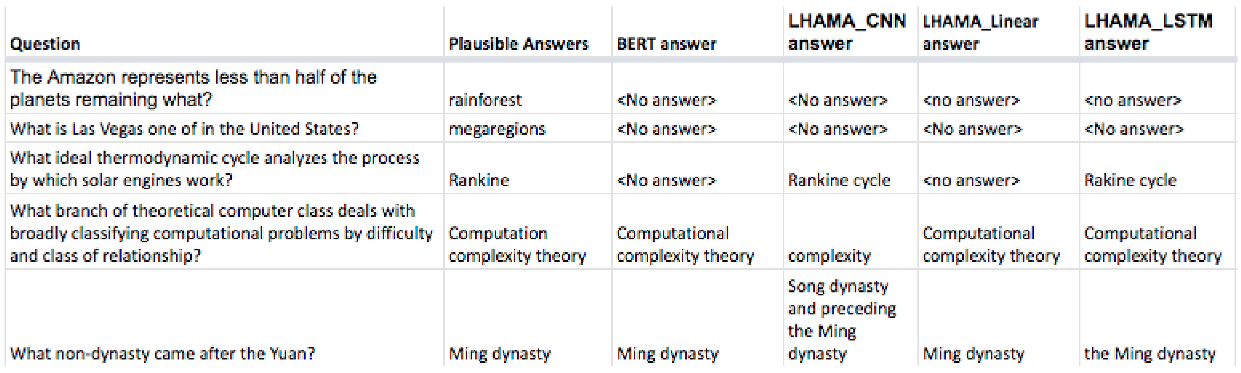
5 Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R’emi Louf, Morgan Funtowicz, & Jamie Brew, ”HuggingFace’s Transformers: State-of-the-art Natural Language Processing,” arXiv:1910.03771 [cs.CL], Oct. 2019.

6 Eric Wallace Pedro Rodriguez, Shi Feng, Ikuya Yamada, & Jordan Boyd-Graber, ”Trick Me If You Can: Human-in-the-loop Generation of Adversarial Examples for Question Answering,” arXiv:1809.02701 [cs.CL], Sep. 2018.

7 Yicheng Wang & Mohit Bansal, ”Robust Machine Comprehension Models via Adversarial Training,” arXiv:1804.06473 [cs.CL], Apr. 2018.

8 Shaojie Bai, J. Zico Kolter & Vladlen Koltun, ”An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling,” arXiv:1803.01271 [cs.LG], Mar. 2018.

A Appendices



Question	Plausible Answers	BERT answer	LHAMA_CNN answer	LHAMA_Linear answer	LHAMA_LSTM answer
The Amazon represents less than half of the planets remaining what?	rainforest	<No answer>	<No answer>	<no answer>	<no answer>
What is Las Vegas one of in the United States?	megaregions	<No answer>	<No answer>	<No answer>	<No answer>
What ideal thermodynamic cycle analyzes the process by which solar engines work?	Rankine	<No answer>	Rankine cycle	<no answer>	Rakine cycle
What branch of theoretical computer class deals with broadly classifying computational problems by difficulty and class of relationship?	Computation complexity theory	Computational complexity theory	complexity	Computational complexity theory	Computational complexity theory
What non-dynasty came after the Yuan?	Ming dynasty	Ming dynasty	Song dynasty and preceding the Ming dynasty	Ming dynasty	the Ming dynasty

Figure 4: Qualitative Analysis of 5 Questions

Hyperparameters used during training:

- Processes per node: 1
- Pre-trained model: bert-base-uncased
- Lowercase: True
- Learning rate: 2e-5
- Epochs: 4
- Max sequence length: 384
- Document stride: 128
- Train/eval batch size: 12