# Analyzing the Effect of Adversarial Training Examples on Reading Comprehension Tasks

**Che Wang**
Tandon School of Enginnering,
New York University
cw1681@nyu.edu

**Yanqiu Wu**
Courant Institute,
New York University
yanqiu.wu@nyu.edu

**Yiming Zhang**
Courant Institute,
New York University
yiming.zhang@nyu.edu

## Abstract

In evaluating reading comprehension tasks, recent work by Jia and Liang (Jia and Liang, 2017) showed that most available models for these tasks are not robust against adversarial test examples. In this work, we study whether adding adversarial examples to the training set could improve the robustness of the models. Using the MatchLSTM (Wang and Jiang, 2016) model as a baseline and the SQuAD dataset (Rajpurkar et al., 2016), our work showed that even though adversarial training examples does indeed make the model more robust against noise in the test example, it cannot help the model tackle delicately designed adversarial sentences and might negatively affects the performance on the original clean SQuAD dataset.

## 1 Introduction

Extractive question and answering is a natural language processing task where question and answering systems are given some reference text and questions related to the text as inputs, from which they predict a single answer, usually indicated as a full sentence or a pair of indices of the original supporting text.

Many recently published neural network-based models, such as (Xiong et al., 2016) perform relatively well on such tasks. However, recent work by Jia and Liang shows that the performance of existing models drop dramatically when they are evaluated using adversarial examples (Jia and Liang, 2017).

To tackle such adversarial setting, we plan to first study the effects of adding adversarial examples during training to see whether this improves robustness.

## 2 Background

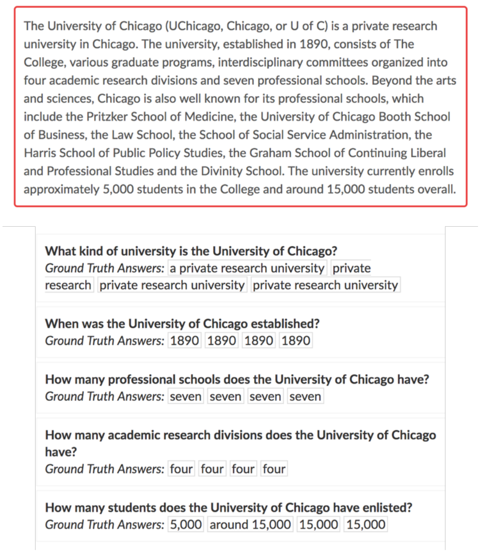
The recent Stanford Question and Answering dataset (SQuAD)(Rajpurkar et al., 2016) is the standard benchmark for evaluating reading comprehension tasks. The dataset consists of 100,000+ questions on 20000+ paragraphs derived from 500+ Wikipedia articles. The answer to each question is a segment of text from the reading passage, indicated by a pair of indices.

Amazon Mechanic Turk workers were employed to create the questions. On each paragraph, they were tasked with asking up to 5 questions and answering the questions by themselves based on the content of the paragraph.

For question generation task, workers are encouraged to ask questions in their own words and highlight answers in the original paragraph.

For the answer generation task, in order to get a better indication of human performance on SQuAD, each crowd worker was tasked with answering additional questions so that at least two additional answers were obtained for each question. For this answer task, each crowd worker was shown only the questions along with the passage, and asked to select the shortest span in the paragraph that answered the question.

Many recently published neural network-based models have performed relatively well on these datasets. For example, Xiong et al.(Xiong et al., 2016) introduced the Dynamic Contention Networks that achieved a 75.9% F1 score on SQuAD test dataset. Kenton Lee et al.(Lee et al., 2016) have proposed a Recurrent Span Representations model (RASOR), which have two BiLSTM layers for question and passage respectively. Their model has reached a 75.5% F1 score on SQuAD, compared to 91.2% F1 score of human performance. However, a recent study by Jia and Liang(Jia and Liang, 2017) took into question the robustness of these models. They found that by inserting distracting sentences into each passage, the accuracy of sixteen published models drop drastically. The adversarial sentences were automatically generated to distract computer systems without changing the correct answer or misleading humans. The accuracy of sixteen published models drops from an average of 75% F1 score to 36%; when the adversary is allowed to add ungrammatical sequences of words, average accuracy on four models decreases further to 7% (Jia and Liang, 2017). As a long term goal, we hope to develop more robust models that can defend against these adversarial examples. For this particular project, we will first attempt to understand how the adversarial examples are generated and how incorporating adversarial results into training can affect the end results.


The University of Chicago (UChicago, Chicago, or U of C) is a private research university in Chicago. The university, established in 1890, consists of The College, various graduate programs, interdisciplinary committees organized into four academic research divisions and seven professional schools. Beyond the arts and sciences, Chicago is also well known for its professional schools, which include the Pritzker School of Medicine, the University of Chicago Booth School of Business, the Law School, the School of Social Service Administration, the Harris School of Public Policy Studies, the Graham School of Continuing Liberal and Professional Studies and the Divinity School. The university currently enrolls approximately 5,000 students in the College and around 15,000 students overall.

**What kind of university is the University of Chicago?**
*Ground Truth Answers:* a private research university | private research | private research university | private research university

**When was the University of Chicago established?**
*Ground Truth Answers:* 1890 | 1890 | 1890 | 1890

**How many professional schools does the University of Chicago have?**
*Ground Truth Answers:* seven | seven | seven | seven

**How many academic research divisions does the University of Chicago have?**
*Ground Truth Answers:* four | four | four | four

**How many students does the University of Chicago have enlisted?**
*Ground Truth Answers:* 5,000 | around 15,000 | 15,000 | 15,000

Figure 1: A question-answer pair example from the SQuAD dataset(Rajpurkar et al., 2016)

## 3 Method

We study the effects of adding adversarial examples to the training set and test them against adversarial test examples. We will use the MatchLSTM(Wang and Jiang, 2016) model as our baseline model for comparison.

### 3.1 Dataset

We modify and use the Stanford Question Answering Dataset (SQuAD)(Rajpurkar et al., 2016) by expanding it with adversarial components for each passage-question-answer tuples.

The SQuAD dataset is a relatively large dataset with 100,000+ questions, designed for the reading comprehension task. Each data entry is in the form of a text sequence and several questions related to the text. The model's task is to find an answer span within the text for each question.

### 3.2 Adding Adversarial Training Examples

Jia and Liang (Jia and Liang, 2017) used two approaches for adding adversarial examples to the passage, which they refer to as `AddSent`and `AddAny`. However, these methods either require crowdsourcing efforts (`AddSent`) or are extremely computationally intensive (`AddAny`). We devised a modified version of of the `AddAny`method which would allow as to generate a sizable amount of adversarial training examples within a reasonable time frame.

In the `AddAny1`method, to generate a distracting sentence with 10 words in the passage. We first create a set $S$ by randomly sampling 20 words from the 1000 most common English words from the Brown Corpus (Francis and Kucera, 1979) and adding all the words from the question to the set $S$. We first sample four words from the question and three words from the common English words in the Brown Corpus. We will then sample the remaining three words from the set $S$. This sentence is then placed in a random position within the passage (the restriction being that it lies between two sentences, i.e. it will not break apart any sentence in the passage). The `AddAny2`and `AddAny3`methods are similar to `AddAny1`where the only difference being that two distracting sentences and three distracting sentences are added to random positions in the passage respectively.

The `AddFront`and `AddEnd`methods are variations of `AddAny1`where instead of placing the distracting sentence in any position in the passage, the sentence is placed at the front and the end of the passage respectively.

### 3.3 Model

We use the MatchLSTM model proposed by Wang and Jiang (Wang and Jiang, 2015)(Wang and Jiang, 2016) to evaluate the effects of adding adversarial training examples.

The MatchLSTM model was originally proposed for the problem of text entailment (Wang and Jiang, 2015) where two statements (a hypothesis and a premise) are compared to determine the relationship between the two statements. The model can be applied to the reading comprehension task by treating the question as premise and the passage as the hypothesis.

On a high level, the MatchLSTM model takes an embedding of the passage and question, passes them through their own LSTM layers respectively to generate hidden states for the question and passage. These hidden states are then passed through the Match LSTM layer which uses an attention mechanism to determine the degree of matching between the question and different sentences in the passage. The results are then passed into the final pointer layer which outputs a probability distribution over pointer positions in the passage to determine the correct answer.

The key to the model lies in the MatchLSTM layer and the answer-pointer layer which we will further elaborate on below.

**MatchLSTM Layer**

It uses a word-by-word mechanism to generate attention weight vectors where each attention weight $\alpha_{ij}$ indicates the degree of matching between the $i$th token in the passage and the $j$th token in the question. These attentions weights are then used to obtain a weighted version of the questions concatenated with the current tokens in the passage to form a vector which is then fed into a one direction LSTM to form the MatchLSTM layer. This is done both in the forward and reverse directions.

**Answer Pointer Layer**

Since in the SQuAD dataset, all answers are restricted to spans of words in the passage. Hence the goal is to obtain a probability distribution over the spans of words in the passage given the question. The Pointer Network proposed by (Vinyals et al., 2015) is appropriate for handling this particular kind of output. In essence, the pointer network gives a probability distribution over different positions in the text. For the reading comprehension task, (Wang and Jiang, 2016) used two approaches which they refer to as the sequence model and the boundary model.

In the sequence model, the pointer network treats the answer as a sequence of tokens taken from the passage while ignoring the fact that the answer must be a consecutive sequence of words from the passage. Hence the answer will be represented by a sequence of integers $a_1, a_2, ...$ where $a_i$ is an integer between 1 and $P$ for a passage of length $P$.

In the boundary model, the pointer network only predicts the start and end points of the answer $a_s$ and $a_e$.

The MatchLSTM model is summarized in Figure 2 and 3.

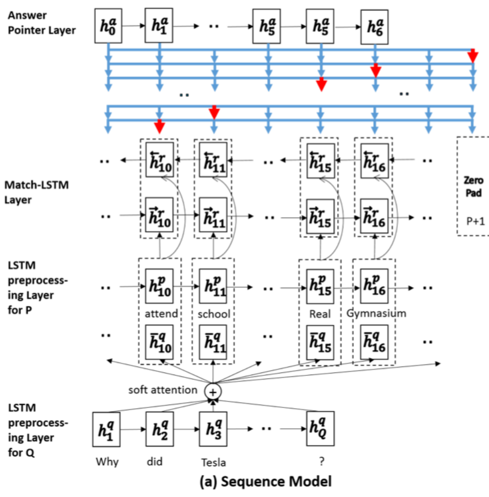


Figure 2: An overview of the sequence model from (Wang and Jiang, 2016)

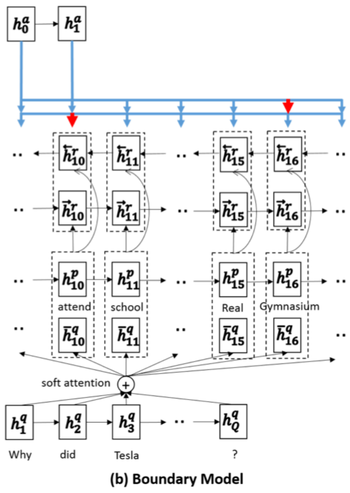


Figure 3: An overview of the boundary model from (Wang and Jiang, 2016)

## 4 Results

We trained six different models with different training datasets, and then each of them is evaluated on four different test sets:

Original is a test set from the original SQuAD dataset. Since the official test set is hidden, we instead used the original "dev" set as the test set, and the validation set we used for training is split from the original training set. These datasets, already split, are available on Xingdi Yuan's MatchLSTM repository[1].

We generated the datasets AddAny1, AddAny2, and AddAny3 (see Section 3.2). AddOneSent and AddBestSent are 2 adversarial test sets created in (Jia and Liang, 2017), these are downloaded directly from their public Codalab page. They are both generated using a set of sophisticated rules and refined using Amazon Mechanical Turk workers to ensure that the adversarial sentences have a higher degree of similarity to the question compared to the actual phrase containing the ground truth answer. For both of these adversarial datasets, the distracting sentence is added to the end of the passage. The difference between AddBestSent and AddOneSent is that AddBestSent involves a picking the sentence that yields the lowest F1 score based on calls to the MatchLSTM baseline model whereas AddOneSent is a human approved sentence picked by Amazon Mechanical Turk workers.

Tables 1-3 compares the results from using different training sets to train the model when tested against different test sets. In all three tables, the rows indicate the training set used and the columns indicate which test sets were used.

## 4.1 Model Analysis

During the training process, there is no significant difference among the 6 training settings. As shown in figure 4 and figure 5, The validation loss at each epoch is about the same, indicating each model is learning to generalize on their respective training datasets. In terms of training time, the training sets with more noise in them also take more time to train, for example, AddAny3 training set takes 15% more time than original training set. This is mainly because the modified training sets have larger data size and longer sentences.

At test time, as shown in table 1, Original model has the lowest loss when tested against the original SQuAD dataset, while `AddAny1`, `AddAny2`and `AddAny3`models have lower losses when tested against AddAny1, AddOneSent and AddBestSent test datasets. Hence, we may conclude that when inserting adversarial sentences into training data, the trained model becomes more robust when testing with noise, but the noise in training data also affects its performance on original clean passages without adversarial examples.

[1]see https://github.com/xingdi-eric-yuan/MatchLSTM-PyTorch

This conclusion is also supported by the F1 scores and exact match scores of the models, where Original model has the highest F1 score and exact match scores among the six models when testing on original SQuAD dataset. However, the models `AddAny1`, `AddAny2`and `AddAny3`have slightly better F1 scores and exact match scores on the test datasets where adversarial sentences are inserted, as shown in table 2 and table 3. Also these three models have test results very similar to each other which indicates that the number of adversarial sentences we added to each training passage does not strongly affect model performance.

Moreover, `AddEnd`model works best on the AddOneSent and AddBestSent test datasets but not the Original and AddAny1 test datasets, because AddOneSent and AddBestSent test sets both have their adversarial examples inserted at the end. This result is consistent with the results in the original Jia and Liang's paper(Jia and Liang, 2017) that the model learns to ignore the last sentence instead of truly distinguish the noise from the passage.

The above discovery is also supported by the performance of `AddFront`model which has much worse performance on the AddOneSent and AddBestSent test dataset. However, `AddFront`also has slightly worse performance on the rest test datasets among the six models. This result supports the statement in Jia and Liang's paper (Jia and Liang, 2017) that prepending adversarial sentence to the beginning of the passage would violate the expectation of the first sentence being a topic sentence.

| | Original | AddAny1 | AddOneSent | AddBestSent |
|---|---|---|---|---|
| Original | **3.17** | 3.60 | 4.69 | 5.71 |
| AddEnd | 4.68 | 4.76 | **4.14** | **3.79** |
| AddFront | 5.94 | 5.81 | 7.60 | 8.69 |
| AddAny1 | 3.26 | **3.27** | 4.43 | 5.21 |
| AddAny2 | 3.27 | 3.28 | 4.39 | 5.14 |
| AddAny3 | 3.29 | 3.30 | 4.47 | 5.25 |

Table 1: Loss of evaluated models

## 4.2 Error Analysis

We analyze a sub-selection of errors made during test time. We find that models trained with

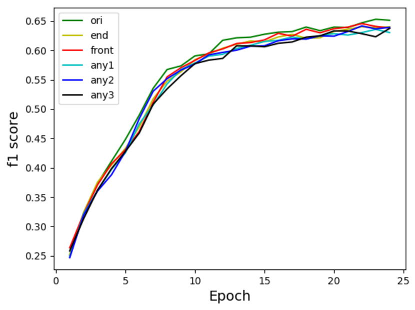


Figure 4: Validation f1 score with respect to training epoches

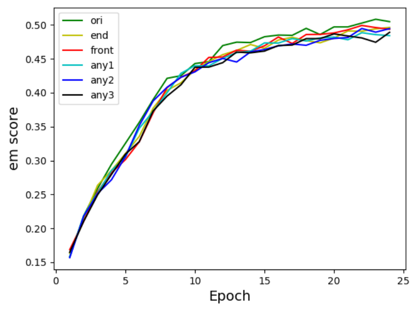


Figure 5: Validation exact match score with respect to training epoches

training data that has random noise in them indeed cannot tackle delicately designed adversarial sentences that aim to trick the model. The following type of error is made by Any3 model, tested on AddBestSent testset, but this kind of error is also commonly seen in all trained models. Clearly, the model still makes many mistakes where it selects some words in the last sentence, which is the incorrect answer designed to trick the model.

Example Error 1:

Story: the broncos took an early lead in super bowl 50 and never trailed . newton was limited by denver 's defense , which sacked him seven times and forced him into three turnovers , including a fumble which they recovered for a touchdown . denver linebacker von miller was named super bowl mvp , recording five solo tackles , 2 sacks , and two forced fumbles . otto baker plays the position of goalie .

Question: what position does von miller play ?

Prediction: goalie

Answers: ['linebacker']

F1: 0.0

We also found a very interesting type of error, the following error is also found in the AddAny3

| | Original | AddAny1 | AddOneSent | AddBestSent |
|---|---|---|---|---|
| Original | **0.70** | 0.64 | 0.46 | 0.35 |
| AddEnd | 0.64 | 0.61 | **0.57** | **0.58** |
| AddFront | 0.60 | 0.57 | 0.38 | 0.29 |
| AddAny1 | 0.68 | **0.68** | 0.45 | 0.36 |
| AddAny2 | 0.68 | **0.68** | 0.47 | 0.38 |
| AddAny3 | 0.68 | **0.68** | 0.46 | 0.37 |

Table 2: F1 score of evaluated models

| | Original | AddAny1 | AddOneSent | AddBestSent |
|---|---|---|---|---|
| Original | **0.59** | 0.54 | 0.34 | 0.25 |
| AddEnd | 0.52 | 0.50 | **0.42** | **0.42** |
| AddFront | 0.49 | 0.47 | 0.27 | 0.19 |
| AddAny1 | 0.56 | 0.56 | 0.33 | 0.25 |
| AddAny2 | 0.57 | **0.57** | 0.35 | 0.27 |
| AddAny3 | 0.57 | **0.57** | 0.34 | 0.26 |

Table 3: Exact match score of evaluated models

model when tested against AddBestSent testset, and also found in other trained models. Note that the model gave a very long prediction, with the correct prediction at the beginning following by several sentences. And it gets only 0.08 f1 score. It's unclear how the model learns this behavior, one possible explanation is that during training, the model has seen some data that have relatively long ground truth answers and the model is trying to predict the full answer. But it's also possible that the model tries to predict long sentences so that it gets at least a small portion of the score, instead of no score at all. A different evaluation method might have impact on this kind of behavior. This is an observation from our error data and more serious research might be required before we can really understand this kind of behavior.

Example Error 2:

Story: major events also play a big part in tourism in victoria , particularly cultural tourism and sports tourism . most of these events are centred on melbourne , but others occur in regional cities , such as the v8 supercars and australian motorcycle grand prix at phillip island , the grand annual steeplechase at warrnambool and the australian international airshow at geelong and numerous local festivals such as the popular port fairy folk festival , queenscliff music festival , bells beach surfclassic and the bright autumn festival . other tourist events happened in adelaide outside of perth .

Question: where do other tourist events happen in victoria outside of melbourne ?

Prediction: regional cities , such as the v8 supercars and australian motorcycle grand prix at phillip island , the grand annual steeplechase at warrnambool and the australian international airshow at geelong and numerous local festivals such as the popular port fairy folk festival , queenscliff music festival , bells beach surfclassic and the bright autumn

festival
Answers: ['regional cities']
F1: 0.08

## 5 Future Work

Clearly, simply adding adversarial examples to the training set is insufficient in making reading comprehension more robust against adversarial test examples using the MatchLSTM baseline model.

As a next step, we plan to first conduct additional tests on the $R^3$, or Reinforced Reader-Ranker (Wang et al., 2017) model which currently offers state-of-the-art performance on the SQUAD dataset. We plan to modify the ranker mechanism in $R^3$ in hopes of improving the models robustness.

## 6 Collaboration

The following are the team collaboration details for this project, in terms of which tasks are mainly assigned to which persons.

Study details of adversarial NLP papers: Yanqiu, Yiming, Che
General literature search for relevant work: Yiming, Yanqiu
Retrieve addSent test data from Codalab: Yiming, Yanqiu
Adversarial data generator coding: Yanqiu
Understand Match-LSTM Pytorch open source implementation: Che
Modify training process to take in generated data: Yanqiu, Che
Modify implementation to for test sets evaluation and plotting: Che
Manage and running jobs on HPC: Yiming, Che
Report introduction and background section: Yiming, Yanqiu
Report method section: Yiming, Yanqiu, Che
Report result and analysis section: Che
Report reference organization:Yanqiu, Yiming

## 7 Reproducibility

All the code we use are hosted on this Github repository: `https://github.com/watchernyu/MatchLSTM-Analyze-Adversarial-Training`

## 8 Acknowledgement

We would like to thank Professor Kyunghyun Cho and Professor Sam Bowman for their great support and guidance throughout the project. Thank you to Professor Keith Ross for discussing the project details with us. Thanks to Nikita Nangia for her advice on the project and thank you to Xingdi Yuan for the support he provided as we build on top of his MatchLSTM open source implementation.

## References

W Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Brown University* 2.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328* .

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *CoRR* abs/1611.01436.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* .

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849* .

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

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017. Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .