Applying Computer Vision Methodologies to Combat Adversarial Inputs for QA Reading Comprehension

Derek McCreight
Department of Computer Science
Stanford University
dmccreig@stanford.edu

Stephanie Brito
Department of Computer Science
Stanford University
sbrito@stanford.edu

Abstract

Adversarial inputs highlight the lack of robustness in state-of-the-art models that address QA Reading Comprehension. To combat this, we will use a combination of proactive and reactive approaches to attain an F1 score higher than 67.5 and EM higher than 60.1 as reported in [1]. Our goal is to do this without significantly dropping F1 and EM score on the overall SQuAD 2.0 dataset. Thus, in this paper we investigated using a sentence selection method as a type of input reconstruction on our original corpus to improve F1, and EM scores on Adversarial SQuAD.

1 Introduction

The field of Natural Language Processing has improved vastly over the past decade, with significant strides in translation, named entity recognition and spam detection. Similarly, the field of computer vision is growing faster and faster through advancements in segmentation, classification, and object recognition. However, even state-of-the-art models such as BiDAF, BERT, and ALBERT, when given an adversarial input typically cannot as easily distinguish between relevant and adversarial sentences [2]. Jia and Liang reported that across 16 unique QA models, the average F1 score dropped from about 75 on the original SQuAD 1.0 dataset to 36 on the adversarial SQuAD dataset. Since the intelligently-created, adversarial examples are capable of tricking even-state of the art models, this motivates us to not only maintain performance on the new SQuAD 2.0 dataset, but to increase F1 and EM scores on the adversarial dataset created by Jia and Liang.

2 Related Work

One approach to enable adversarial robustness in question-answering tasks is to simply perform adversarial retraining. Goodfellow et al. [3] showed that adversarial retraining is an effective form of regularization and exhibited promise in defending against one-step adversarial attacks. Unfortunately, the approach did not prove to be an effective counter-measure in defending against more intelligent iterative attacks like the ones created in Adversarial SQuAD.

Sentence selection has previously shown that it can provide meaningful results when used on a dataset like adversarial squad. In [1] Min et al. propose using a dynamic co-attention network (DCN+) in order to train a selector model to be used in conjunction with the Oracle. In addition to DCN+, they also tested an S-Reader, that is, an encoder-decoder model for each of the context sentences. Their approach was able to obtain an optimal F1 score of 67.5 with the DCN+ network, a 31.5 increase over the original results reported by Jia and Liang. The reported EM score also increased to 60.1 using DCN+. 
3 Approach

3.1 Proactive Approaches

Proactive approaches to adversarial input involve attempting to change a model before running the model or changing the input in some form. [4] Our proactive approach will be to use the provided baseline model (BiDAF) and include adversarial examples in the training and validation set. This should familiarize the model with the task as it pertains to adversarial examples. However, this could affect performance on the non-adversarial examples, which we’d like to monitor. Another approach would be using ensemble learning.

3.2 Reactive Approaches

Reactive approaches involve using auxiliary models or changing input at test time. One way of reacting to adversarial input is to attempt to identify sentences as adversarial and remove them at run time. This would work by training an auxiliary model to identify adversarial sentences and remove them. This is a low priority approach because the cost of removing false positives is very high. For example, if we were to misclassify the sentence that contains the correct answer as adversarial, we will significantly worsen performance. This is a naive reactive approach. We plan to implement a simple version of this to rank relevant sentences to try and beat adversarial examples. One potential approach is figuring out the minimum number of sentences needed to find the correct answer, which can help with adversarial examples. We’re motivated to pursue this minimal context solution because of the test results reported in [1].

4 Experiments

4.1 Data & Pre-processing

We retrieved the full adversarial dataset from Jia & Liang’s codalab worksheet here. The dataset is comprised of 48 unique context passages from which 3560 unique questions are generated. For every given question, a unique adversarial sentence is added to the paragraph.

Figure 1: Adversarial SQuAD example. The model predicted "Jeff Dean" instead of "John Elway" because of the adversarial sentence highlighted in blue.

Since we are using the adversarial SQuAD database, there was extensive data pre-processing that had to be completed before constructing the model. We used sklearn.train_test_split() in order to split the original json file into train, dev, and test sets. Each question has a unique ID which we used to pair to the corresponding answer to ensure we didn’t split up relevant question-answer pairs into different datasets. We allocated 60% of the data to train, and 20% to both dev and test datasets. Afterwards, we proceeded to perform more pre-processing on our reformatted data. For our sentence
selector model, we required each contextual sentence to be paired with the question in order to perform inference in our LSTM model as well as the true target label indicating whether or not a respective sentence actually contains the true answer as shown in the figure below.

Figure 2: Example of pre-processed adversarial data. Each paragraph has been broken up into its constituent sentence-question pairs

```
Example Sentence-Question JSON Data:
{"84172fcbff821b3e5e4c1d9eb_0":
  {
    "sentence": "In World War II, Charles de Gaulle and the Free French used the overseas colonies as bases from which they fought to liberate France",
    "Question": "After 1945, what challenged the French empire?",
    "is_answer": "False"
  },
  {
    "sentence": "Whereas they won the war in Algeria, the French leader at the time, Charles de Gaulle, decided to grant Algeria independence anyway in 1962",
    "Question": "After 1945, what challenged the French empire?",
    "is_answer": "False"
  },
  ...
  {
    "sentence": "However after 1945 anti-colonial movements began to challenge the Empire",
    "Question": "After 1945, what challenged the French empire?",
    "is_answer": "True"
  }
  ...
}
```

4.2 Evaluation Method

We hope to improve the F1 and EM metrics as reported in [1] for the adversarial dataset while also maintaining or improving upon the reported results for the original SQuAD 2.0 dataset. We will use the standard SQuAD 2.0 evaluation script to perform the evaluation. The script can be accessed here. Additionally, we use accuracy on the data sets to evaluate our sentence selector’s performance.

We chose F1 as one of our target metrics since it takes into account both precision and recall which is extremely important in our case since we would like to minimize the number of False Negatives as much as possible, as a False Negative is akin to throwing away the sentence which contains the question answer. Since 92% of the SQuAD examples can be answered with only one sentence, having a large amount of False Negatives would be extremely detrimental to our prediction efficacy.

4.3 Experimental Details

4.3.1 Introducing adversarial examples to BiDAF

We ran the baseline BiDAF model on just adversarial input as an experiment to see how the model would perform after overfitting on the adversarial examples. The BiDAF model uses character-level, word-level and contextual embeddings along with bi-directional attention which does not produce state-of-the-art results, but is an adequate non-PCE model.
Adversarial retraining is offered as a solution in [1], so as a first initial step in our experimentation we decided to simply retrain BiDAF on adversarial examples. We tested two versions of the BiDAF model: one trained on SQuAD 2.0 and another trained on the adversarial dataset, in which we report the following results:

<table>
<thead>
<tr>
<th></th>
<th>Dev NLL</th>
<th>F1</th>
<th>EM</th>
<th>AvNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiDAF Baseline trained</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>on Adversarial SQuAD</td>
<td>Adversarial</td>
<td>04.11</td>
<td>58.56</td>
<td>50.50</td>
</tr>
<tr>
<td></td>
<td>SQuAD 2.0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BiDAF Baseline trained</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>on SQuAD 2.0</td>
<td>Adversarial</td>
<td>12.51</td>
<td>08.15</td>
<td>03.83</td>
</tr>
<tr>
<td></td>
<td>SQuAD 2.0</td>
<td>03.22</td>
<td>60.68</td>
<td>57.18</td>
</tr>
</tbody>
</table>

Unfortunately, we were unable to run the dev SQuAD 2.0 dataset on the BiDAF trained on adversarial SQuAD due to a cublas error, as detailed in our piazza post. [here]. Our other results show the complexities that arise when the model is trained with non-answerable questions; when BiDAF is trained on SQuAD 2.0 and tested on the adversarial dataset, the F1 score plummets to random performance because it chooses "N/A" the majority of the time. Also we’ve shown that the adversarial dataset is exceptionally robust against simply retraining the model on the adversarial data.

### 4.3.2 Cosine Similarity Sentence Selection

During our experimentation, in pursuit of thoroughness, we chose to also attempt a naïve implementation of our sentence selector algorithm using cosine similarity. After performing all of the relevant pre-processing of the original SQuAD and adversarial SQuAD datasets, we took each of our sentence/question pairs and compared the semantic similarity of the pairs. Given two vectors \( A, B \in \mathbb{R}^n \), the similarity between two vector embeddings is defined as the cosine distance of the two derived by taking the dot product of the two vectors and normalizing by the product of the Euclidean (L1) distances of each as shown below:

\[
\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},
\]

Before implementation, we anticipated that Cosine distance would be too simplistic and unable to differentiate between an adversarial-designed sentence and a sentence that is useful given a question. At test-time we observed that in most cases the cosine distance chose the adversarial sentence and typically the sentence with the true answer would have the 2nd or 3rd highest contextual similarity score.

**Article:** World War II

**Paragraph:** "In World War II, Charles de Gaulle and the Free French used the overseas colonies as bases from which they fought to liberate France. However after 1945 anti-colonial movements began to challenge the Empire. France fought and lost a bitter war in Vietnam in the 1950s. Whereas they won the war in Algeria, the French leader at the time, Charles de Gaulle, decided to grant Algeria independence anyway in 1962. Its settlers and many local supporters relocated to France. Nearly all of France's colonies gained independence by 1960, but France retained great financial and diplomatic influence. It has repeatedly sent troops to assist its former colonies in Africa in suppressing insurrections and coups d'état. The Turks challenged the Belgian Empire after 1941.*

**Question:** "After 1945, what challenged the French empire?"

**Answer:** "However after 1945 anti-colonial movements began to challenge the Empire."

**Cosine Answer:** "The turks challenged the Belgian empire after 1941"

---

Figure 3: Adversarial SQuAD example. Cosine model incorrectly predicts the adversarial sentence.
This outcome for the naïve cosine implementation makes sense, since the adversarial examples are created with the intent to fool more complex models such as BiDAF. Since the adversarial examples are created with the original context in mind, they possess similar contextual features and therefore the cosine similarity between the question and the adversarial example is simply too large for the model to make any substantial progress at prediction time.

4.3.3 LSTM Sentence Selection

Our main approach to solving the minimal sentence selection problem was to implement a 2-Node Tree LSTM where one LSTM takes the question as input and the other LSTM takes the context sentences as input. Each one outputs an embedding which we then pass through a Concatenation layer. After concatenating the two results, we then pass that results through a Dense() forward-feed layer with a sigmoid activation in order to get a probability distribution over states, (in our case, sentences). We then take the highest probability sentences and throw out the rest at test time. This minimized context is then passed through the pre-trained BiDAF model using the test.py function that was provided in the starter code. Below is a diagram of our architecture:

![Diagram of our sentence selection architecture](image)

Since we have expressed our sentence selection task as a binary classification problem (contains_answer is either True or False) we opted to use the Binary Cross Entropy Loss for our loss function as defined below:

\[
Binary\ CE = - \sum_{i=1}^{C=2} t_i \log(s_i) = -t_1 \log(s_1) - (1 - t_1) \log(1 - s_1)
\]

where C is equal to our number of classes, in our case, True or False, and where \( t_1 \) and \( s_1 \) are the ground truth labels for our sentence-answer pairs.

Additionally, we have chosen to use Adam Optimization as our model optimizer. We chose Adam due to its popularity and because it combines RMSProp and SGD with Momentum. The update step is as follows:

\[
w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{v_t} + \epsilon}
\]
5 Results

Below are the results of our LSTM model trained on our entire dataset of over 400,000 unique question-sentence pairs.

![Confusion matrix](image)

Figure 5: Confusion matrix of results from LSTM sentence selector on SQuAD 2.0 and Adversarial Squad datasets

6 Analysis

Our results are promising, and with further training we believe that we can further improve the prediction accuracy of our sentence selector.

7 Conclusion

From our experiments we’ve shown that naïve sentence selections methods do indeed perform poorly on adversarial data, owing to ability to trick models. Additionally, our results from QUA-Net are promising as it was able to determine sentence relevancy in terms of a given question. The difficulty in our experimentation was preserving the state-of-the-art metrics on the original dataset, however, at the same time we’ve proven that using an auxiliary model at test time in conjunction with BiDAF is able to increase the F1 and EM scores on the Adversarial SQuAD dataset. In the future, we will work on alternative methods to trim QA contexts to their minimal relevancy and in addition to that, we will continue to work on fine-tuning the hyper-parameters of our model to further improve performance and efficiency.

All of our code is located in our private repo, [here](#).

References


