Analyzing and Attempted Fixing of Data Artifacts in SQuAD Dataset

Jason Chan Jin An (jjc5439)
jasonchan@utexas.edu

Abstract

With the rapid advancement in Natural Language Processing (NLP), the NLP community has seen a rise in the amount of large and complex pre-trained model. The leading NLP startup behind pre-trained models is Huggingface, who has a strong NLP community behind it alongside its open-source library Transformers and large repository of pre-trained models. This reduces the barrier of using pre-trained models and benchmarking on NLP tasks. However, are these pre-trained models really "solving" the tasks that the dataset encapsulates? In this paper, the SQuAD dataset is used to analyze data artifacts learned by the model by modifying some data with adversarial examples and *Checklisting*. The results are benchmarked using the ELECTRA small model and examples that achieve a really low score are further analyzed for features that contains the data artifacts. These issued are then attempted to be mitigated with methods such as adversarial training and data augmentation. The effectiveness of these solutions are also further analyzed and evaluated as to why they do or do not work. Lastly, a recap of the analysis and fixes are concluded.

1 Introduction

Using pre-trained models in deep learning for NLP tasks is now a common practice with the exponential growth and advancements in pre-trained models. Utilizing pre-trained models can be beneficial and leverage on the weights and knowledge from training over similar task. Furthermore, pre-trained models are easy to implement and integrate in the pipeline of the task. In some cases, pre-trained models substantially improves the quality of various complementary model components (Hendrycks et al., 2019). This could very well be the case when evaluating test sets. However, how would one know if these good results are actually coming from the model being able to learn from the encapsulated datasets? Or if the model has learned spurious correlations or data artifacts and hence being able to recognize these spurious patterns that are coincidentally predictive on the test set?

In this paper, analysis is done using the **SQuAD dataset** (Rajpurkar et al., 2016) and the **ELECTRA small model** (Clark et al., 2020). There are two parts to this work, the first is the analysis part and the second part the fixing part. The analysis part focuses on showing the situations the model is expected to fail by changing the data in some ways such that data artifacts are not present. This is shown by adding Adversarial Examples in the context (Jia and Liang, 2017), which is meant to distract the model without modifying the actual answer or misleading humans. Another data changing method would be using Checklist Sets (Ribeiro et al., 2020) where comprehensive behavioral testing of NLP models is conducted on the SQuAD dataset with taxonomy and robustness tests where some question answers are Checklist generated and some are augmented squad data. The failure rate is then computed for each test.

As for attempting to fix these data artifacts, training on adversarial examples of the dataset was attempted in hopes to make the model recognize adversarial examples and not be so easily swerved by it. The effectiveness of this fix is then further evaluated on modified adversarial examples. Lastly, a more complex model and data augmentation of the dataset was also attempted.

2 Experiments Approach and Structure

The structure of the experiments in this paper is structured in a way such that **the analysis of each data artifact problem are first explored followed by its attempt to fix this problem.** In

this way, the effects of the problems can be isolated and analyzed in detail while the effectiveness of the fix can be gauged in a more coherent manner. Therefore, the structure is as follows:

Analysis 1: Exploring effects of adversarial sentences on performance on SQuAD dataset metrics.

Analysis 1 fix: Training on adversarial examples and exploring the effectiveness.

Analysis 2: Evaluating the model using Checklist tests and its performance.

Analysis 2 fixes: More complex model and data augmentation.

3 Analysis 1: Effects of Adversarial Sentences

To analyze the effects of adversarial sentences, an adversarial sentence is added at the end of the context of each sample in the dataset. This process is known as **AddSent**. The raw SQuAD dataset with AddSent is taken from the CodaLab repository [1] of (Jia and Liang, 2017) under the name *train-convHighConf.json*. The ELECTRA model (trained on context without adversarial sentences, also known as the **baseline model**) is then used to predict on the AddSent dataset. The results are then recorded and compared with the baseline ELECTRA results on unmodified SQuAD data.

3.1 Results and Insights

Below are the results of the different datasets the ELECTRA model used for evaluation and prediction. The dataset split proportion into train, validation, and test is 80:10:10. The main evaluation metrics used here are **Exact Matching (EM) and F1 Score**.

Train Data	Eval Data	Train EM	Train F1
Ordinary	Ordinary	79.9	90.0
Ordinary	AddSent	30.8	35.4

Table 1: Results of training set EM and F1 Score.

Train Data	Eval Data	Val EM	Val F1
Ordinary	Ordinary	67.5	80.6
Ordinary	AddSent	29.4	37.2

Table 2: Results of validation set EM and F1 Score.

[1] https://worksheets.codalab.org/worksheets/0xc86d3ebe69a3427d91f9aaa63f7d1e7d/

Train Data	Eval Data	Test EM	Test F1
Ordinary	Ordinary	67.8	80.3
Ordinary	AddSent	30.8	35.4

Table 3: Results of test set EM and F1 Score.

From the 3 tables above, it is evident that ELECTRA baseline model trained on ordinary SQuAD dataset does not perform well when asked to predict on AddSent instances. This means that the adversarial sentences managed to fool the model into skewing its predictions. Upon examining the predictions on the validation and test sets, the instances where the model can actually predict the answer correctly and incorrectly are analyzed.

Successful Prediction Insights

Upon further inspection on cases where the baseline model predicts the answer correctly despite an adversarial sentence at the end, it is discovered that this is usually the case when the **adversarial sentence is unrelated or has less lexical overlap to the question or context.** Consider an example below where the adversarial sentence is in green and gold answer in red:

Context: *St. Anselm's* approach was to define God as, "that than which nothing greater can be conceived". Famed pantheist philosopher Baruch Spinoza would later carry this idea to its extreme: "By God I understand a being absolutely infinite, i.e., a substance consisting of infinite attributes, of which each one expresses an eternal and infinite essence." For Spinoza, the whole of the natural universe is made of one substance, God, or its equivalent, Nature. His proof for the existence of God was a variation of the Ontological argument. *Chicago defined god as " that than which nothing lesser can be conceived.*

Question: Who defined god as "that than which nothing greater can be conceived"?

Actual Answer: st. anselm
Baseline Answer: st. anselm

From the example in the text box above, it is obvious that the adversarial sentence at the end is about Chicago whereas the context is about God. Therefore, making it unrelated and hence the baseline model can still predict the right answer.

Unsuccessful Prediction Insights

On the other hand, it is discovered that the baseline model fails to predict the right answer when the **adversarial sentence is somewhat related or has high lexical overlap to the question or context.** Consider an example below where the adversarial sentence is in green and gold answer in red:

Context: One advantage of the black box technique is that *no programming knowledge is required.* Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested. *Aliens is one huge disadvantage to using the black-box method.*

Question: What is one huge advantage to using the black-box method?

Actual Answer: no programming knowledge is required
Baseline Answer: aliens

The adversarial sentence in the example above clearly has high lexical overlap because it contains the keywords *"huge disadvantage to the black-box method"*. These keywords have high lexical overlap with the question and context. Hence, fooling the baseline model.

4 Analysis 1 Fix: Training on Adversarial Data

In an attempt to fix the low scores obtained in the previous section, the lowest hanging fruit would be to train a new ELECTRA the AddSent SQuAD dataset (coined as **AddSent model**). This hopefully could enable the model to learn to recognize adversarial sentences and produce the right answer. It is also important to make sure the AddSent model can predict the non-adversarial context correctly. Therefore, this will be evaluated as well.

Lastly, to gauge the true effectiveness of the trained AddSent ELECTRA model, the model is used to predict 1,000 **AddSentMod** examples. These are similar to AddSent but uses different fake answer candidates and appends the adversarial sentence to the beginning of the paragraph, instead of appending it to the end.

4.1 Results and Insights

Below are the results of the different datasets the AddSent model used for evaluation and prediction. The dataset split proportion into train, validation, and test is 80:10:10. The main evaluation metrics used here are **Exact Matching (EM) and F1 Score**. The final table is the AddSent model's prediction results on the AddSentMod examples.

Train Data	Eval Data	Train EM	Train F1
AddSent	Ordinary	61.6	74.0
AddSent	AddSent	87.9	93.7

Table 4: Results of training set EM and F1 Score.

Train Data	Eval Data	Val EM	Val F1
AddSent	Ordinary	53.1	66.5
AddSent	AddSent	75.9	84.9

Table 5: Results of validation set EM and F1 Score.

Train Data	Eval Data	Test EM	Test F1
AddSent	Ordinary	53.4	66.6
AddSent	AddSent	76.6	85.1

Table 6: Results of test set EM and F1 Score.

Train Data	Eval Data	Test EM	Test F1
AddSent	AddSentMod	30.8	42.4

Table 7: Results of the 1,000 AddSentMod examples EM and F1 Score.

Based on the results above, there are a few interesting observations. The AddSent model does very well when used to evaluate the AddSent data. However, it seems that the AddSent model performs relatively worse on ordinary (unmodified) SQuAD dataset. Furthermore, it also does not do well on the AddSentMod data. Upon further inspections, it is discovered that the AddSent model did not learn to recognize adversarial sentences, **but rather just ignores the last sentence.** This is

evident especially in the AddSentMod data where adversarial sentences are added in the beginning of the context as well. This implies that training a model on AddSent data **does not make the model more robust to adversarial examples (unless the adversarial sentence happens to be the last sentence).**

An obvious example of this would be below where the actual answer is in red and adversarial sentence is in green:

Context:
The Champ Bowl 40 took place in Amsterdam. Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at *Santa Clara, California.* As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

Question: Where did Super Bowl 50 take place?

Actual Answer: 'Santa Clara, California'
AddSent Model Answer: amsterdam

Despite using the AddSent model, the model still gets fooled as the adversarial sentence was appended at the start. It seems that the model has just learned to ignore the last sentence instead.

5 Analysis 2: Model Analysis with Checklist

In the previous section, effects of adversarial sentence appended to the end of the context was explored. However, this is just one of the tests that can be done to explore anymore potential data artifacts learned by the model. A more robust method to generate more of these examples has be developed by (Ribeiro et al., 2020) under the name **Checklist** [2]**.**

The analysis for this part is done by first loading the *Huggingface* pipeline with the trained ELECTRA model (**baseline model**) on unmodified SQuAD dataset. Checklisting is mainly done on the **context of the data** and answers predicted using the baseline model. The failure rates for each test case are recorded. The Checklist category tests that were selected are **Taxonomy, Negation, and Robustness.** The tests will use a mix of Checklist generated data and also SQuAD data.

5.1 Results and Analysis

Below are the results of the Checklist tests using the baseline model. **A sample of 500 is taken from the validation set for each test.**

Data (Sample of 500)	Category	Test	Failure Rate (%)
Checklist generated	Taxonomy	Nationality and Professions	100
Checklist generated	Taxonomy	Synonym	100
Checklist generated	Negation	Negation	100
Checklist generated	Robustness	Temporal	100
SQUAD validation	Robustness	Change names	26
SQUAD validation	Robustness	Add random sentence	0
SQUAD validation	Robustness	Typo	0

Table 8: Results of Checklist tests using baseline model.

From the results table above, it seems that the baseline model performs poorly for most tests. In fact, it has a 100% failure rate for the first four tests. This implies that the baseline ELECTRA model has did not learn synonyms, negation, temporal, nationality, and profession in its weights.

As for the final 2 tests of add random sentence and typo, it seems that the failure rate is 0%. However, this can be misleading as **Checklist defines a failure as the predicted answer not changing, rather than actually predicting the correct answer.** Therefore, this just means that the predicted answer by the baseline model is not affected by

[2] https://github.com/marcotcr/checklist

typos or adding random sentence. Perhaps the model is not complex enough and hence not sensitive to these tests.

6 Analysis 2 Fixes: More Complex Model and Data Augmentation

6.1 More Complex Model

The first would be to try and use a more complex model. As the baseline ELECTRA model is actually taken from Huggingface's *google/electra-small-discriminator* repository, it only has 12 layers, 256 hidden nodes, and 14M parameters. Therefore, a more complex model, which is the *distilbert-base-cased-distilled-squad* model is then used and subjected to the same Cheklist tests in the section above. The BERT model is definitely more complex as it has 6-layer, 768-hidden nodes, 12-heads, and 65M parameters. Therefore, a valid assumption would be the BERT model would be more robust to the Checklist tests and hence should perform better than the baseline model.

The results of the BERT model are as follows:

Data (Sample of 500)	**Category**	**Test**	**Failure Rate (%)**
Checklist generated	Taxonomy	Nationality and Professions	20.2
Checklist generated	Taxonomy	Synonym	18.4
Checklist generated	Negation	Negation	90.9
Checklist generated	Robustness	Temporal	1.0
SQUAD validation	Robustness	Change names	6.6
SQUAD validation	Robustness	Add random sentence	14.2
SQUAD validation	Robustness	Typo	16.2

Table 9: Results of Checklist tests using the more complex BERT model.

The BERT model seems to have much lower failure rates for nationality and professions, synonym, temporal and change names tests. This means that the model is has already learned to recognize these tests and predict the right answer as it is complex enough to do so. However, it seems that the BERT model also does not do well is recognizing negation, which is expected as the SQuAD dataset does not really have this problem to learn from.

Another main difference is that the BERT model does have some failure rates in the last two tests, which are adding random sentence and typo. This means that there are instances where adding a random sentence or a typo would change the the predicted answer of the BERT model. With all things considered, it is safe to come to a conclusion that using a more complex model does improve robustness towards checklist tests.

6.2 Data Augmentation

Another method to fix this would be to try to augment the SQuAD dataset and introduce some elements of the Checklist tests into it. Functions from the checklist Python library were used to randomly induce elements of the checklist tests on the training set's context of the SQuAD dataset. A new ELECTRA model (coined as **augmented model**) can be trained on this augmented dataset and then used to go through the checklists tests again. The results from the augmented model can then be compared with the results above.

Similarly, a sample of 500 was taken and the results of the augmented model are as follows:

Data (Sample of 500)	**Category**	**Test**	**Failure Rate (%)**
Checklist generated	Taxonomy	Nationality and Professions	57.6
Checklist generated	Taxonomy	Synonym	7.9
Checklist generated	Negation	Negation	80.0
Checklist generated	Robustness	Temporal	0.4
SQUAD validation	Robustness	Change names	9.0
SQUAD validation	Robustness	Add random sentence	23.4
SQUAD validation	Robustness	Typo	14.8

Table 10: Results of Checklist tests using the augmented ELECTRA model.

The augmented model has improved performance is all the checklists tests. In addition, it also has some failure rates for the last 2 tests, implying the augmented model is more sensitive to typos and adding random sentence. With these results, it is evident that training on induced checklist test augmented data can improve robustness to

these tests even by just using an ELECTRA small model.

7 Conclusion

In this paper, the extent of data artifacts in the SQuAD dataset and attempts to fix it was explored via a series of analysis and fixes. The first analysis explored was the effects of adding adversarial sentences to the end of the context (AddSent data). Using a baseline model did not do well as it was mostly "tricked" by the adversarial sentence when it has high lexical overlap with the question or context. An attempted fix to this was training an adversarial model (by training on examples with adversarial sentences at the end of the context). The results seems promising at first as the adversarial model performed very well on the AddSent data. However, when tested against AddSentMod data, it is discovered that the adversarial model has simply learned to ignore the last sentence rather than actually recognizing adversarial sentences.

The second analysis of data artifacts by using Checklist tests on the SQuAD data. The baseline model was first used in the Checklists tests and did not perform well in most of the test cases. It was also not sensitive to typos and adding random sentences. An attempt to fix it was by using a more complex model which is the BERT model. The BERT model was able to give reduced failure rates of the test cases and showed increased sensitivity towards typo and adding random sentences. Lastly, data augmentation was done on the SQuAD training set and an augmented ELECTRA model was trained and put through the Checklist test. The augmented model was able to obtain reduced failure rates and increased sensitivity to typos and adding random sentences.

All in all, data artifacts are hard to combat but depending on the use case and how the problem is framed, using a more complex model and data augmentation can help mitigate it. However, more robust testing is also important to truly gauge the effectiveness of these fixes.

Acknowledgments

I would like to personally thank the lecturer, Greg Durett and the teaching assistants of this course as I have learned so much in this course in terms of the fundamentals and deepened my understanding in this area. Also a huge shout-out to teaching assistants for responding to the class' questions in a timely manner.

References

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. 2019. Using pre-training can improve model robustness and uncertainty. *CoRR*, abs/1901.09960.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *CoRR*, abs/1707.07328.

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

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Association for Computational Linguistics (ACL)*.