# Adapting CheckList to ELECTRA for QA: Fine-Tuning for Challenge Datasets

**Anonymous CS388 submission**


## Abstract

Several challenges have been made to the SQuAD dataset over time. These include issues with robustness, biases in the training data, and lack of "difficult" or multi-step question answering. Challenge datasets such as Adversarial SQuAD (Jia and Liang, 2017) and Adversarial QA (Bartolo et. al, 2020) have been proposed to strengthen Question Answering models. Additionally, testing methods such as CheckList (Ribeiro et. al, 2020) have been designed to expose weaknesses in existing datasets and testing methodologies for QA models. The goals of this paper are three-fold: first, to adapt CheckList tests to a SQuAD-trained ELECTRA QA model (Clark et. al, 2019) to understand its weaknesses and strengths. Second, to train it using adversarial datasets and evaluate which categories improved. Lastly, attempting to use hand-tuned training sets to further strengthen performance on specific types of tests from CheckList. I will show that while it is possible to improve performance using hand-tuned sets on certain categories, other categories may also see either increased or reduced performance. Even though a "one-size-fits-all" QA model was not achieved in this experiment, high performance on specific types of questions is demonstrated.


## 1 Introduction

Question Answering (QA), while historically a task given to computers as early as the 1960s with a Baseball-specific question/answer machine (Green et. al, 1961),[1] it has come into the mainstream within the last decade, and the SQuAD dataset was introduced at Stanford in 2016 (Rajpurkar et. al, 2016) to enable widespread development and training of QA models on over 100,000 labeled examples.

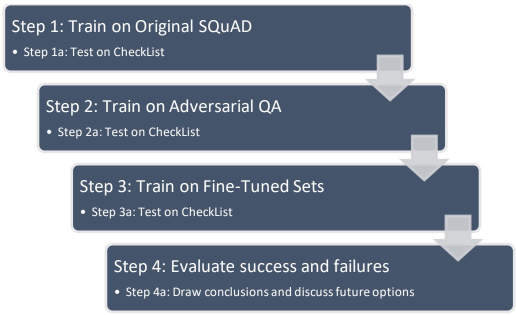


Figure 1: Illustration of general flow of experiment, utilizing adversarial datasets like those of Jia and Liang (2017) and inspired by fine-tuning methods of Liu et. al (2019).

However, over time, challenges have been introduced to models developed and trained using open-source datasets like SQuAD, exposing weaknesses and areas with room for improvement (Jia and Lang, 2017; Rajpurkar et. al, 2018; Bartolo et. al, 2020). An important part of understanding and improving models such as those used for Question Answering is knowing the limitations of the models: what they can do, and more importantly what they cannot do. For example, SQuAD-trained QA models have not been shown to demonstrate "multi-hop" question answering (Chen and Durrett, 2019), as questions in the dataset do not demand "multi-hop" reasoning such as those in the HotpotQA dataset (Yang et. al, 2018).

I studied the literature referenced above and others, and chose to train the ELECTRA model using the Huggingface Transformers framework (Wolf et. al, 2019), utilizing starter code provided in the assignment template. Initial training was done using the 'train' split of the SQuAD dataset. Evaluation was performed by adapting the

[1] In hindsight it seems obvious that baseball statistics nerds would be one of the earliest to design a computer system to query their mountains of stats.

CheckList tests to work with the provided code and HuggingFace Transformer framework. CheckList suggests various test categories, including Vocabulary, Taxonomy, Robustness, Temporal, Negation, Coreference, and Semantic Role Labeling (SRL); each is evaluated using percentage of tests failed (presented in the paper later on as percentage of tests passed). Further training was conducted by introducing data from adversarial QA challenge sets to inoculate the model (Liu et. al 2019), such as AddOneSent and AddSent from the Adversarial Examples paper's `squad_adversarial` dataset (Jia and Lang, 2017) and `adversarial_qa` dataset (Bartolo et. al, 2020). I initially hypothesized improvements could be made to CheckList test category performance using only samples of adversarial QA data for training, but this was not consistently backed by results. This motivated me to pursue fine-tuning training using small datasets to improve performance on specific test types.

I compared the percentage performance of each testing category from CheckList over various combinations of SQuAD and adversarial data attempts and attempted to train the model on specific hand-made datasets to help the model learn the concepts of negation and former/latter relationships. Both categories were routinely scoring approximately 0.0% regardless of underlying dataset and training. Training data was adapted from the CheckList tests, but specifically randomizing sentence structure, names, professions, and answers and keeping the negation and former/latter relationships. This data, after one training pass, increased performance from 0% passing to upwards of 60%, as well as incidental improvements on other categories of CheckList tests. In most cases of adversarial and small dataset training, F1 performance on the standard SQUAD dataset was improved from my initial baseline, and in all cases never significantly diminished in performance. Finally, I also evaluated the various trained models on the adversarial_qa model from Bartolo et. al, 2020 and saw significant improvement from the base SQuAD-trained ELECTRA model, but only an increase to approximately 40% F1; nothing close to state-of-the-art performance.

I show in this paper that significant performance improvements can be made to specific query types using fine-tuning and CheckList on an ELECTRA model for Question Answering, with minimal loss of generalization for other types of queries and minimal overall performance impact on general SQuAD F1 performance.

# 2 Methodology

## 2.1 Initial Setup/Hardware Configuration

This experiment was conducted primarily on Google Colab with "GPU" and "premium GPU" options, in order to maximize performance during computation-intensive training and evaluation phases. Colab Pro+ subscription was utilized to minimize disconnect from runtime, which would have negatively impacted the performance of the experiments. Python version 3.7.15 was used due to an incompatibility between the default Colab Python 3.8 environment and the CheckList testing suite; this downgrade was achieved by selecting "Use fallback runtime version" from the Colab Command Palate.[2]

## 2.2 Initial Training and Evaluation

I started with a "stock" ELECTRA model from the HuggingFace Transformers pipeline and trained it using default parameters on the SQuAD database to achieve my "baseline" results. The first evaluation was conducted by evaluating the model on exact match and F1; receiving 78.01% and 86.09, respectfully.

From there, I downloaded and modified the CheckList testing package from `https://github.com/marcotcr/checklist`. I modified the SQuAD loading starter code provided in the CheckList repository in order to properly format the SQuAD test package as a JSONL datafile that Huggingface Datasets could easily import with existing code (see Appendix A for function details). My initial plan was to develop my own tests using the CheckList categories but after spending multiple days attempting to understand and implement these tests using the included packages, unfamiliarity

[2]This option will not be available after "mid-December" 2022, according to Google Colab documentation: https://github.com/googlecolab/colabtools/issues/3246

with Python and Huggingface Pipelines prevented me from making full use of the CheckList suite (including `Editor` and `TestSuite` functions). After significant effort was expended, I changed course and ended up falling back on the provided TestSuite which runs over 79,000 prebuilt tests on the categories in the CheckList paper (Ribeiro et. al, 2020).

The baseline SQuAD model performed well on certain CheckList suite test categories and failed significantly on many others; the details will be shown in the "Results" section below. The subsequent training was based off either the "stock" ELECTRA model or this SQuAD-trained model, referred to "baseline model," or "baseline" in the following sections.

## 2.3 Subsequent Testing and Evaluation Methodology – squad_adversarial

My next goal was to run several experiments to determine how the ELECTRA model could be enhanced to perform better on CheckList suite tests. I ran experiments to train the model on two different Adversarial QA datasets: `squad_adversarial` (Jia and Lang, 2017) and `adversarial_qa` (Bartolo et. al, 2020) under various subsets and combinations of these datasets.

First, I trained the "stock" model on the AddOneSent subset of the squad_adversarial dataset combined with SQuAD dataset via Huggingface `Datasets.combine_dataset()` functionality and noted slight improvement on exact match % and F1, as well as improvement on Taxonomy type tests from CheckList but minimal improvement or slight decreases on others.

Next, I trained the stock model on the AddSent subset of the `squad_adversarial` dataset combining with SQuAD dataset; the goal was to introduce the model to more examples of adversarial QA data in hopes of improving performance on SQuAD data and CheckList tests, as AddSent contains more examples of adversarial sentences. This resulted in another slight improvement to exact match % and F1 on SQuAD and adversarial_qa validation sets, as well as subsequent improvements on some Robustness and Taxonomy tests, but no significant improvements or changes on most other types of tests. I decided I needed to investigate a second adversarial QA dataset.

## 2.4 Testing and Evaluation Methodology – adversarial_qa

From there, I investigated the `adversarial_qa` dataset created by Bartolo et. al 2020, and decided it might add some more robustness to the model. I conducted two different experiments with this dataset; first performing subsequent training on the "baseline" SQuAD-trained model with this challenge set, as well as using combine_dataset to combine this dataset with AddSent of squad_adversarial dataset.

Conducting follow-on training of the baseline model with challenge data, as opposed to combining the challenge data with the SQuAD dataset to train the "stock" model, showed some surprising results. This includes improving significantly on certain CheckList sections like comparison and taxonomy, and the ability to distinguish "his/her" relationships at a 48% rate, but also a significant performance drop in categories that other experiments retained or improved, including Robustness, Named Entity Recognition, and Fairness. Due to the uneven results, this model of "follow-on" training of the stock model was not continued for my subsequent experiments.

The next experiment was combining both adversarial challenge datasets with the original SQuAD dataset and conducting full training cycle on this combined "triple" dataset. This dataset showed the strongest performance of all experiments on exact match % and F1; 80.49% and 87.54, respectively. It also showed similar or improved performance on nearly every CheckList test category to the baseline SQuAD-trained model, with only one category decreasing more than five percentage points (his/her distinction). Due to these results, I decided to adopt this "triple" trained model as the new baseline; subsequent references to this model will be noted as "triple" or "triple dataset" model.

## 2.5 Testing and Evaluation – Fine-Tuning

Taking the "triple" dataset trained model, I was motivated to investigate whether or not specific categories of CheckList tests could be approached by training the model further on a small fine-tuned dataset with specific examples.

Specifically, my goal was to train the model on the relationship between “former/latter” and named entities earlier in the passage, and the concept of negation of earlier statements/descriptors. I approached each one separately, measured performance on overall SQuAD data and CheckList tests, and then attempted training with a combined fine-tuned dataset with both categories.

I noted that previous models showed close to 0.0% accuracy on the Former / Latter relationship portion of CheckList test suite and, motivated by the Inoculation by Fine-Tuning paper (Liu et. al, 2019), I decided to adapt small training sets (approximately 100 entries) with explicit negation in both the context and the question to evaluate whether the model could successfully “learn” this relationship. I also created a dataset of approximately 100 entries with explicit former / latter relationships. In both fine-tuned datasets, I randomized names, adjectives, descriptors, etc. using the following sites: in an attempt to prevent the model from learning specific words as features when the tests were run. I first performed follow-on training with this small former/latter dataset on the “baseline” SQuAD trained model, noting an increase from 0.0% of tests passing to only 6.1% passing, but no significant decreases in performance on overall SQuAD data or other CheckList categories. Due to these results, I then decided to train the “triple” trained model with the former/latter dataset and noted marked improvements in both former/latter recognition (42.95% accuracy) and significant progress in Taxonomy and Fairness category tests, as well.

I then attempted to train the model on the former-latter dataset for a second time, but this resulted in slight decreases across the board, including even a slight decrease in “former/latter” test performance so I did not pursue this approach. The success on the single training cycle of “former/latter” fine-tune dataset motivated me to create the negation dataset mentioned above and evaluate the results of training the “triple” model on another small dataset to fine-tune results.

I created the dataset of approximately 125 examples to train the model on examples of negation: specifically including passages with negation and questions with negation, as well as samples with negation only mentioned in the question. I found that the triple dataset evaluated very strongly on this category after just training on a small set of examples, as high as 99.38% accuracy on this subset of tests. While I did randomize names, descriptors, professions, etc., to reduce chance of over-fitting the model it is possible that the model was still able to easily over-fit this training data and I will discuss this possibility in results and conclusions below. After evaluating the model on negation, I wanted to combine the two datasets and attempt to train the model on both concepts: negation and former/latter.

I then combined the two datasets and trained the “triple” model on a former/latter and negation fine-tuning set and evaluated results on SQuAD and CheckList. Overall, the fine-tuned model performed similarly to the “triple” and “baseline” models on overall SQuAD performance, losing only about 1 point off F1 from the triple dataset and still 1 point higher than the baseline, but showed significant improvement in several categories of CheckList tests, not just negation and former/latter. Specific results and conclusions as to why will be drawn later in this paper.

### 2.6 Resources – Scripts, Notebook, Datasets, Results

All datasets used, training scripts modified, and the Jupyter notebook I ran in Google Colab, as well as results received are listed below:

https://anonymfile.com/y6kBy/results.xlsx
https://anonymfile.com/OaX2Q/run.py
https://anonymfile.com/1yPBV/requirements.txt
https://anonymfile.com/KV2Q7/nlp-final.ipynb
https://anonymfile.com/0mPob/f1-eval-results-for-various-models.txt
https://anonymfile.com/znmaj/helpers.py
https://anonymfile.com/DXNm9/squad-negation-2.jsonl
https://anonymfile.com/xJroy/squad-negationformerlatter.jsonl
https://anonymfile.com/2pWa1/squad-formerlattertrain.jsonl
https://anonymfile.com/n051a/squad-formatted.jsonl
https://randomwordgenerator.com/name.php
https://www.randomready.com/random-job-generator/

**Results by Category on SQuAD Validation Sets and Selected CheckList Tests**

| Training Datasets | SQuAD validation (exact % / F1) | Adversarial QA validation (exact % / F1) | Vocab Comparison/ Synonyms | Taxonomy attributes/ nationality-job/animal-vehicle | Fairness Male-Female Stereo-types | Temporal Change in job | Negation | Coreference His-her | Former -Latter |
|---|---|---|---|---|---|---|---|---|---|
| SQuAD | 78.01%/86.09 | 19.93%/30.09 | 1.21%/97.09% | 1.40%/35.40% / 36.35% | 56.52% | 99.59% | 6.02% | 6.20% | 0.00% |
| SQuAD + AddSent | 80.24%/87.32 | 18.07%/28.01 | 0.81%/98.88% | 1.00%/63.20%/ 38.86% | 8.00% | 84.23% | 0.00% | 0.20% | 0.00% |
| SQuAD then AddSent | 69.92%/78.54 | **29.23%/39.61** | 45.95%/84.79% | **69.20%/79.40%** /52.71% | 4.60% | 98.76% | 0.81% | **48.20%** | 0.00% |
| SQuAD + adversarial + AddSent (Triple) | **80.49%/87.54** | 28.47%/39.00 | 2.02%/95.75% | 10.80%/40.80% /48.09% | 55.10% | 96.47% | 4.49% | 1.00% | 0.00% |
| SQuAD then former-latter | 77.23%/84.69 | 21.03%/29.26 | 1.01%/92.39% | 4.40%/32.60%/ 34.34% | 6.95% | 8.30% | 0.20% | 37.60% | 6.11% |
| Triple then former-latter set | 79.60%/86.39 | 28.00%/37.87 | 10.12%/44.07% | 37.40%/47.00% /37.25% | 25.25% | 6.85% | 0.00% | 39.80% | 42.95% |
| Triple then negation set | 80.04%/87.00 | 28.30%/38.21 | 64.78%/97.32% | 6.20%/19.80%/ **74.20%** | **100.00%** | **100.00%** | **99.38%** | 0.06% | 0.00% |
| Triple then negation+ former-latter set | 79.46%/86.58 | 28.00%/37.34 | **78.34%/97.76%** | 16.60%/6.20%/ 70.08% | **100.00%** | 99.59% | **99.79%** | 4.40% | **60.21%** |

Table 1: Performance of experimental models on SQuAD and various CheckList test categories. Not every CheckList category is included due to spacing; see the full spreadsheet at `https://anonymfile.com/y6kBy/results.xlsx` for more details.

Note that most benchmarks are achieved on either the SQuAD-trained "baseline" followed by subsequent training on adversarial data, or on the "triple" set trained further on negation/former-latter datasets.

## 3 Results

Selected results are shown in Table 1; all else in the spreadsheet hyperlinked in Table 1's caption.

### 3.1 Results from Baseline SQuAD training

Training the stock ELECTRA model on SQuAD 'train' split dataset resulted in results of 78.01% exact match and 86.09 F1. Evaluation on the 'validation' set of the adversarial_qa dataset (Bartolo et. al, 2020) resulted in only 19.93% exact match and 30.09 F1. This closely reproduces the results of Jia and Lang (2017) in that strong performance on the standard SQuAD is not nearly robust enough to perform well on human-created adversarial examples.

Results from Baseline model on CheckList tests enabled several observations, as well. The initial SQuAD dataset enables the model to pass the Vocabulary test category of "Synonyms" well, as this is well-represented in the question/answer set. I achieved 97.09% passing on CheckList Synonyms just with the baseline SQuAD training data. However, the next two Vocabulary tests, Comparison and Intensifiers failed, passing only 1.21% and 0.00% of tests, respectively. This motivated me to examine the SQuAD dataset and I found that neither of these two examples were represented well in the underlying data. While comparison is something that performed better upon using fine-tuning datasets later on, the datasets I used did not improve performance on Intensifiers and I would have liked to investigate separately if I had more time.

Baseline SQuAD model showed mixed results on Taxonomy category; failing Size/Shape/Age/Color tests with 1.40% passing, and reaching only 35.40% and 36.35% respectively for distinguishing Nationality from Profession and distinguishing Animals from Vehicles. These categories showed drastic improvement when training on challenge data, discussed later. The baseline model also failed on Antonym tests and Coreference tests, but none of the models achieved significant success on these categories.

Additionally, the SQuAD-trained baseline model performed very well on Robustness tests and Named Entity Recognition, passing Question Typo with 75.20%, Question Contractions with 88.00%, Adding Random Sentence with 84.60% (this one specifically was the highest of all models). Changing Name and Changing

Location tests passed with 91.60% and 84.20% respectively.

Mixed results were achieved on Temporal category tests, with the baseline model passing 99.59% of "Change in Profession" temporal tests but passing 0.00% of "before/after to first/last" tests. None of the models achieved success on the "before/after to first/last" categories and this is also something I would investigate if fine-tuning could help if I had more time.

Semantic Role Labeling was something that the baseline model, as well as subsequent models, largely failed to perform on and deserves further investigation.

Finally, the baseline model achieved poor performance on Negation tests and Former / Latter tests. The baseline model achieved 6.02% on Negation tests and 0.00% on Former / Latter tests; this was part of my motivation to fine-tune on these test cases.

### 3.2 Results from Adversarial training

Training on various adversarial datasets, including those from Jia and Lang (2017) and Bartolo et. al (2020) resulted in insignificant changes in performance when only `squad_adversarial` data was combined with original SQuAD data, but showed significant results when the model was trained with `squad_adversarial` as a second set following initial baseline SQuAD training, as well as when both `squad_adversarial` and `adversarial_qa` datasets were combined together with SQuAD dataset.

Overall, when `AddSent` or `AddOneSent` were inserted into the SQuAD dataset, the F1 performance on both SQuAD and adversarial_qa validation sets were minimally impacted; less than 1.5 points each. CheckList tests were minimally impacted as well; however, ability to distinguish Nationality from Profession in the Taxonomy category did increase from 35.40% in the baseline to 63.20% with AddSent data included. It became clear that just training with adversarial data mixed into SQuAD data would not be enough to make significant progress on either the SQuAD F1, the adversarial_qa F1, or CheckList test categories. This conclusion motivated training with `squad_adversarial` dataset as a secondary training set instead of mixed-in to see if values could be optimized further from SQuAD baseline training.

I then trained the baseline SQuAD-trained model on the `squad_adversarial` AddSent dataset as a secondary dataset. This resulted in a sharp drop in F1 performance on SQuAD data, similar to that shown by Liu et. al in 2019, but in fact significant performance changes in evaluation on `adversarial_qa` validation set (39.61 F1, the highest achieved of any model tested), and several CheckList test categories. First, the model gained approximately 45% ability to distinguish Comparison tests, and succeeded the best of any model tested on several Taxonomy tests, including Size/Shape/Age/Color and Profession vs. Nationality; 69.20% and 79.40% each, respectively. Coreference performance improved drastically, increasing from 6.20% to 48.20% as well. This model trained on adversarial data following SQuAD data did not fare as well on Male/Female stereotypes, Robustness tests, or Named Entity Recognition tests; but overall it did not lose more than approximately 10-15 percentage points on any; this approach shows promise for those categories (Taxonomy and Comparison, as well as Coreference) if one is willing to take the performance hit on other categories The overall performance on SQuAD decreasing by about 9 points F1 meant that I did not utilize this method in further experimentation, but I would certainly investigate more if I had more time or larger scope.

After abandoning separate dataset sequential training, I then combined both adversarial sets with SQuAD data and trained the stock model on this combined dataset, referred to earlier and below as the "triple" dataset. This achieved the highest performance on exact match and F1 (80.49% and 87.54 respectively), as well as second highest F1 on `adversarial_qa` set (39.00 vs the previous method's 39.61). Additionally, this model did not lose more than 1-3 percentage points of performance on any CheckList category from the baseline SQuAD trained model, and gained anywhere between 5-15% performance increases on many CheckList categories. This increased performance may be attributable to the variance in questions and variance in passages utilized by combining different data sources. I chose to continue with this as the primary model moving forward due to the promising performance shown. From here, my goal was to approach specific CheckList test categories and attempt to improve performance on those questions without loss of generality on SQuAD, `adversarial_qa`, and the other CheckList test categories.

### 3.3 Results from Fine-Tuning training

Overall results show that performance can be sharply increased on specific test categories using small (~100 entry) datasets fine-tuned to the task given, without large general loss in overall performance or on any given category of specific CheckList test.

As described in methodology, the first attempt was training the baseline SQuAD model on a former-latter challenge set; this showed slight decreases in generalized performance (exact match and F1), as well as on various CheckList test categories, such as Fairness and Temporal decreasing from 56.52% -> 6.95% and 99.59% ->8.30%, respectively. It only significantly gained in Semantic Role Labeling (Agent/Object), increasing to 19.32% from 9.46% as well as Coreference His/Her, increasing to 37.60% from 6.20%. Even Former/Latter relationship tests only increased to 6.11% from the baseline result of 0.00%. This was discouraging to see, but I was motivated to try this dataset on the "triple" trained model.

When trained further with fine-tuning sets like former-latter and negation, the "triple" trained model performed strongly on both general performance and specific categories. For example, the triple-trained model with former-latter training achieved 42.95% passing on the former/latter test, but also significant increases in performance in Coreference his/her, Taxonomy tests. Performance did decrease on Fairness and Temporal; this merited further investigation and in fact performance was restored when training with the addition of the "negation" dataset.

Training the triple model on the negation challenge set showed promising results, sharply increasing performance on Comparison tests, Animal vs. Vehicle Taxonomy, Male/Female stereotypes, Change in profession, and Negation tests (passing the last item with 99.38%). Such a high rate of passing might be attributed to overfitting; I tried to minimize this by randomizing professions, descriptors, names, and other words in the tests but due to the overall structure remaining similar it may not generalize to larger passages. Since both negation and former/latter relationships require locality, though (the two sentences with the relationship will still be next to each other, in the same relative order), I am optimistic that it would still apply in longer, more general passages. My next step in a further experiment would be to analyze this performance further to determine if the data was overfit, and if so, how general performance on these types of questions (beyond CheckList) fares.

Overall, combining the two fine-tuning sets (Negation set and Former/Latter set) and performing additional training with the combination on the "triple" trained model showed benchmark or near-benchmark performance on several categories: Vocabulary Comparison, Comparison-Antonyms, Male/Female Stereotypes, Negation, Temporal Change in Profession, Former / Latter (60.21%), and was within 1-3 points of the majority of every other CheckList category. Additionally, it was still ahead in F1 performance on both the SQuAD validation set and the `adversarial_qa` validation set. It remains to be seen whether all these specific results can be generalized to larger datasets, but I can confidently state that test categories from CheckList can be trained with fine-tuning without loss of generality on overall performance.

### 3.4 Discussion

As mentioned above multiple times, it is not entirely clear that training on the fine-tuned datasets will generalize to larger datasets. Investigating whether other categories of CheckList tests can be fine-tuned as well as attempting to test the model on more varied passages to evaluate CheckList category performance would be the next logical step for this experiment, but it would be out of scope (and time) for a single person project in this context.

The fact that performance benchmarks were reached in some categories using both adversarial training data and fine-tuning data speaks to the promise of each method. I also draw the conclusion that a QA model can be adapted to specific types of questions more successfully; if a QA model is going to be used to answer Taxonomy type questions, it may be worth a tradeoff of loss of generality on other categories to improve performance on that specific type of question. Perhaps a higher model could even be trained to determine type of question asked, which could then be fed to a lower QA model specifically trained on those types of questions/features. This idea is far out of the scope of this project, but is an interesting thought experiment.

Another conclusion drawn via my observations is that the “Male/Female stereotype” test was hugely influenced by training data and approach. This helped me realize how sensitive these types of models are to biased input, and the importance of those working on commercial models being aware of the propensity for bias in complex neural models. If models are being updated consistently with new data/search queries, the teams must be equally watchful for bias or decreased performance on certain categories occurring.

Finally, none of the training I did impacted performance on Intensifiers, asking questions with opposite Antonyms, “He/She” coreferences, or Semantic Role Labeling with three agents/objects in the sentence. I would pursue fine-tuning on these test categories, but the fact that there was not any incidental improvement from adversarial data or fine-tuning other categories, as many other CheckList test categories showed, might mean that these types of linguistic patterns are more difficult to adapt to an ELECTRA SQuAD-trained model.

I remain optimistic that Former/Latter and Negation features will be similar in larger more varied datasets than those evaluated in the CheckList release package, as they are very specific structures and orders of sentences, but I would be very interested to develop more complex context/question pairs to test these categories in a further project and rule out over-fitting.

Overall, these results demonstrate that CheckList can be a valuable tool to analyzing performance of a Question-Answering model, and that specific types of bugs can be fixed or performance improved by using Fine-Tuning to teach the model those features.

## 4 Conclusion

Training an ELECTRA QA model on SQuAD followed by adversarial data and fine-tuning using small datasets can illuminate performance issues with current QA models and datasets. While “passing” performance was not achieved on an adversarial dataset, significant improvement was shown in F1 score (from 30.09 to as high as 39.61). Improvement was made on several categories of test from the CheckList suite, without loss of performance on the vast majority of CheckList category or significant decrease to overall F1 performance on SQuAD validation set. A few categories of CheckList tests were not affected by any adversarial training or fine-tuning; it remains to be seen if those categories can be fine-tuned or if they are a more complex challenge than a SQuAD-trained ELECTRA model can address. The incidental performance increase in other categories of CheckList tests may be attributed to features of the CheckList suite questions being too similar between categories, or due to over-fitting of data. Nevertheless, the methods show promise and deserve evaluation over more complex testing datasets, likely derived from CheckList or another evaluation method.

Additionally, male/female and other built-in “biases” are an important topic that should continue to be researched further to avoid potential negative externalities of these types of biases as “AI” and “Machine Learning” continue to become buzzwords and complex models that may not be fully understood by those using them get implemented into everyday systems like job hiring processes.

Overall, the process of training on adversarial data and fine-tuning on small datasets shows promise for improving performance of SQuAD-trained ELECTRA models and potentially other types of complex Question Answering models.

## Acknowledgments

I would like to acknowledge my wife, who actually has a published scientific paper to her name (Theisen et. al, 2017), and thus indulged me by making me feel more important about this project/report than I maybe should have.

## References


Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the AI: Investigating Adversarial Human Annotation for Reading Comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678.

Jifan Chen and Greg Durrett. 2019. Understanding Dataset Design Choices for Multi-hop Reasoning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4026–4032, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In International Conference on Learning Representations.

Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference. ACM, pages 219–224.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019. Inoculation by Fine-Tuning: A Method for Analyzing Challenge Datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2171–2179, Minneapolis, Minnesota. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Frances Theisen, Rebecca Leda, Vincent Pozorski, Jennifer M Oh, Nagesh Adluru, Rachel Wong, Ozioma Okonkwo, Douglas C Dean 3rd, Barbara B Bendlin, Sterling C Johnson, Andrew L Alexander, Catherine L Gallagher. 2017. Evaluation of striatonigral connectivity using probabilistic tractography in Parkinson's disease. Neuroimage Clin. 2017 Sep 9;16:557-563. doi: 10.1016/j.nicl.2017.09.009.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pier-ric Cistac, Tim Rault, R'emi Louf, Morgan Funtow-icz, and Jamie Brew. 2019. Huggingface's trans-formers: State-of-the-art natural language process-ing. ArXiv, abs/1910.03771.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

# A Jupyter Notebook

Notebook used to train and work with model(s) for NLP F22 final project

```
from google.colab import drive
    drive.mount('/content/drive', force_remount=True)

# change to proper directory
%cd /content/drive/MyDrive/fp-dataset-artifacts
#!git pull origin main
%ls

import os

os.environ['USER'] = 'jackgopack4'
os.environ['PASS'] = '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'
os.environ['REPO'] = 'gregdurrett/fp-dataset-artifacts'
# Clone Prof Durrett's github repo for final project
!git clone https://$USER:$PASS@github.com/$REPO.git # clone the repo
```

Download the release data from CheckList github repository

```
!wget https://github.com/marcotcr/checklist/blob/master/release_data.tar.gz
!tar -xvzf release_data.tar.gz
```

Format the release data from CheckList repo in order to be imported using HuggingFace Datasets

```
import json
def load_squad(fold='validation'):
    answers = []
    data = []
    ids = []
    files = {
        'validation': '/content/drive/MyDrive/fp-dataset-artifacts/release_data/squad/squad.json',
        }
    f = json.load(open(files[fold]))
    i = 0
    for t in f['data']:
        i+=1
        for p in t['paragraphs']:
            i+=1
            context = p['context']
            for qa in p['qas']:
                d = {'id': str(i),'context': context, 'question': qa['question'], 'answers': {'tex
t': list(["","",""]), 'answer_start': list(['0','0','0'])}}
                #print(d)
                data.append(d)
                #print(qa['answers'])
                #answers.append(set([(x['text'], x['answer_start']) for x in qa['answers']]))
    with open(os.path.join('/content/drive/MyDrive/fp-dataset-artifacts/release_data/squad/', 'squ
ad_formatted.jsonl'), encoding='utf-8', mode='w') as f2:
        #f.write(json.dumps(data))
        for d in data:
            f2.write(json.dumps(d))
            f2.write('\n')
    return data
load_squad()

# Select "Use fallback runtime version" in Colab Command Palate
# (only available until mid-december), validate 3.7.15 is version shown
!python3 --version

# install the dependencies
!python3 -m pip install --upgrade pip
!python3 -m pip install -r requirements.txt
```

```
#train the model
!python3 run.py --do_train --task qa --model ./trained_model-combined-3 --dataset ./huggingface/huggingface/datasets/experiments/squad_negation_former_latter.jsonl --output_dir ./trained_model-combined-3-negation-former-latter/
```

Evaluate on CheckList squad data (category-based)

```
# evaluate the model
!python3 run.py --do_eval --task qa --dataset /content/drive/MyDrive/fp-dataset-artifacts/release_data/squad/squad_formatted.jsonl --model ./trained_model-combined-3-negation-former-latter/ --output_dir ./eval_output_checklist_combined-3-negation-former-latter/
```

Evaluate on squad data (overall accuracy)

```
!python3 run.py --do_eval --task qa --dataset adversarial_qa --model ./trained_model-combined-3-negation-2/ --output_dir ./eval_output_adversarial_qa-combined-3-negation-2/

!rm -rf /content/drive/MyDrive/fp-dataset-artifacts/eval_output_og_checklist_full_model/
```

Following code adapted from checklist repo at https://github.com/marcotcr/checklist

```
!jupyter nbextension install --py --sys-prefix checklist.viewer
!jupyter nbextension enable --py --sys-prefix checklist.viewer

#!pip install checklist
import checklist
from checklist.test_suite import TestSuite
import logging
logging.basicConfig(level=logging.ERROR)
suite_path = '/content/drive/MyDrive/fp-dataset-artifacts/release_data/squad/squad_suite.pkl'
suite = TestSuite.from_file(suite_path)
#print(suite.get_raw_example_list()[0:100])

pred_path = '/content/drive/MyDrive/fp-dataset-artifacts/eval_output_og_checklist_combined/eval_predictions.jsonl'
suite.run_from_file(pred_path, overwrite=True, file_format='pred_only')
suite.summary()
#suite.visual_summary_table()
```

# B Exact Match % / F1 Results for all experimental models

```
# Results for squad-trained model:
{'eval_exact_match': 78.01324503311258, 'eval_f1': 86.09164447174685}

# Results for squad combined with AddOneSent trained model:
{'eval_exact_match': 79.91485335856197, 'eval_f1': 87.27578558154276}

# Results for squad with AddSent trained model:
{'eval_exact_match': 80.23651844843897, 'eval_f1': 87.31952701446365}

# combined plus adversarial dataset
{'eval_exact_match': 69.92431409649953, 'eval_f1': 78.54105339971485}

# squad plus adversarial dataset
{'eval_exact_match': 69.2336802270577, 'eval_f1': 78.20658141400816}

# triple dataset
{'eval_exact_match': 80.49195837275307, 'eval_f1': 87.54094922682714}

# triple dataset trained on custom former-latter dataset
{'eval_exact_match': 79.60264900662251, 'eval_f1': 86.38583782036187}

# triple dataset double-trained on custom former-latter dataset
{'eval_exact_match': 79.45127719962157, 'eval_f1': 86.24196078975284}

# squad-trained model with custom former-latter dataset
{'eval_exact_match': 77.22800378429517, 'eval_f1': 84.69200902152039}

# combined-triple dataset with negation experiment set
{'eval_exact_match': 80.33112582781457, 'eval_f1': 87.42694235095531}

# combined-triple dataset with negation-2 experiment set
{'eval_exact_match': 80.0473036896878, 'eval_f1': 86.99738144846238}

# combined triple with negation and former latter experiment set
{'eval_exact_match': 79.46073793755913, 'eval_f1': 86.58063084794101}
```