Stacking Ensemble to Improve ELECTRA-small's Performance on Adversarial SQuAD

Kwan Lee, Ph.D.
Department of Computer Science
University of Texas at Austin
ORCID: 0000-0002-9454-1945

***Abstract* — In this project, the author fine-tuned the ELECTRA-small checkpoints using training data from SQuAD and then evaluated them on Adversarial SQuAD. This was done to test whether the model answers questions based on a true understanding of the context, or merely predicts answers based on dataset artifacts. Results show that the model is more likely to make an error if the question is more similar to the distractor. To improve performance of the trained model, the author attempts to use the stacking ensemble method. He creates five ELECTRA-small Level-0 models with part of training data to evaluate those models on the rest, and those evaluation results again are used as training inputs for the meta learner. Unfortunately, it turned out unsuccessful, presumably because of some design errors and the differences between training and validation data in SQuAD.**

Alternatively, ELECTRA-small is trained on SQuAD V2, which includes questions that cannot be answered in the context. The author suspects SQuAD V2 would weaken the model's tendency to pick answers based on the similarity between the question and the distractor, which is supported.

Keywords* — *QuestionAnswering, ELECTRA, SQuAD, Adversarial SQuAD, SQuAD V2, Ensemble, Stacking

I. Introduction

Question-answering (QA) from text [4,6,8] is considered the next-level reading comprehension challenge compared to previous natural language processing (NLP) tasks such as sentiment analysis and named entity recognition. Even compared to information retrieval (IR), which focuses mostly on locating the relevant information within the context, high-level QA requires proper understanding of the given context and synthesizing necessary information to yield an answer. Although some research shows that existing models boast of high scores of QA tasks, scholars have wondered whether the model actually learns from the context or simply manages to find surface-level correspondences between the question and part of the given text [2,5].

Drawing from the latest NLP research on QA, I used the ELECTRA-small (ELECTRA, hereinafter) checkpoints [1] and trained it on SQuAD [6] to explore how the ELECTRA-small model trained on SQuAD manages to answer questions based on a real understanding of the context rather than relying on dataset artifacts.

To this end, Adversarial SQuAD [5] is a very useful dataset to test whether the model fine-tuend on SQuAD does understand the context and answer the question. It adds a distractor at the end of each context in the original SQuAD and applies the same question in SQuAD. With the introduction of a distractor, testing on Adversarial SQuAD provides a good research setting to investigate if the trained model incorrectly answers the question due to the superficial similarity between the distractor and the question. Results show that the performance of the trained ELECTR-small on Adversarial SQuAD is worse than on the original validation set of SQuAD.

To improve the performance of the model, I attempt to employ stacking, one of the ensemble methods; I first fine-tuned ELECTRA-small with a subset of SQuAD training data and created five base models, and then trained a meta-learner with the rest of the training data. Unfortunately, this process requires a deeper understanding of Hugging Face libraries and the QA model to yield a successful design. As an alternative, I also train ELECTRA with SQuAD V2 instead of the original SQuAD with the assumption that it will reduce the model's reliance on superficial similarity, because it contains examples that is unanswerable, not particularly because SQuAD V2 has more training data.

II. Error Analysis

Validation Set	Summary Statistics	
	Match Rate	***F1***
SQuAD (trained on SQuAD)	76.37	84.67
Adversarial SQuAD (trained on SQuAD)	50.28	57.57
Adversarial SQuAD (no training)	28.1	7.28

TABLE I. VALIDATION RESULTS OF ELECTRA

A. *Overview*

After ELECTRA is fine-tuned on SQuAD's training set, it does pretty well on its own validation set. Its match rate is 76.37 and F1 score is 84.67. However, the same architecture and checkpoint is tested using Adversarial SQuAD (AddSent), the exact match rate is no more than 50.28, and the F1 score stands only at 57.57. In other words, using the adversarial SQuAD as the validation set lowers the match rate by 26.09 and F1, by 27.10. For comparison, the ELECTRA without fine-tuning matches only 28.1% of Adversarial SQuAD with F1 of a mere 7.28.

B. *Analysis of Error Patterns*

The types of errors that ELECTRA made when being tested on Adversarial SQuAD can be classified broadly into two types: specific errors, which are rarely related to the nature of adversarial distractors included in Adversarial SQuAD, and general errors, resulting from distractors in the adversarial dataset. Such general errors will reveal the limitations of the architecture and the training set.

1) *Specific Error 1*

Specific errors occur when the ELECTRA architecture and its learned parameters (weights and biases) fail to correctly answer the question even before a distractor is added to the context. In this first type of such errors, because the question itself is badly constructed, trained ELECTRA yields the different wrong answer. Different distractors slightly perturbed from each other lead to varied answers unrelated to one another. There are several distractors in the following example, but here only one distractor is introduced.

- Question: What objective would be labeled as practical?

  Gold-label answer: skill.

- Context: The objective is typically a course of study, lesson plan, or a practical skill. A teacher may follow standardized curricula as determined by the relevant authority. The teacher may interact with students of different ages, from infants to adults, students with different abilities and students with learning disabilities.

  Predicted answer without a distractor: a course of study, lesson plan.

- Distractor: The objective of that would be labeled as impractical.

  Predicted answer with a distractor: impractical.

2) Specific Error 2

Another case shows that the model correctly answer the question without a distractor. Among several distractors, some distractors manage to confuse the model while others do not pose a barrier to the model. One from each type of distractors are presented below.

- Question: Which Middle Eastern nation in particular views Genghis Khan as a contemptible perpetrator of genocide?
  Gold-label answer: Iran

  Context: In the Middle East, and particularly in Iran, Genghis Khan is almost universally condemned as a destructive and genocidal warlord who caused enormous damage and destruction to the population of these areas. Steven R. Ward wrote that Overall, the Mongol violence and depredations killed up to three-fourths of the population of the Iranian Plateau, possibly 10 to 15 million people. Some historians have estimated that Iran's population did not again reach its pre-Mongol levels until the mid-20th century.

  Predicted answer without a distractor: Iran

- Distractor 1: The East Western nation in particular views Kublai Shah as a estimable perpetrator of genocide of Chicago.

  Predicted answer: The East Western nation

- Distractor 2: The western nations in particular view Kublai Shah as a estimable perpetrator of genocide in Chicago.

  Predicted answer: Iran

3) General Errors

General errors suggest that all distractors added to a certain context succeed in confusing the trained the model. These errors are strong evidence that the trained model does not seem to properly understand the context and the question. The trained model becomes confused by distractors, and it seems that the model predicted an answer based on the sentence-level similarities, comparable to the BLEU score, between the question and the distractor. It shows that the model may not be fully utilizing the context.

a) Example 1

- Question: What was the win/loss ratio in 2015 for the Carolina Panthers during their regular season?

  Gold-label answer: 15-1.

- Context: The Panthers finished the regular season with a 15-1 record, and quarterback Cam Newton was named the NFL Most Valuable Player (MVP). They defeated the Arizona Cardinals 49-5 in the NFC Championship Game and advanced to their second Super Bowl appearance since the franchise was founded in 1995. The Broncos finished the regular season with a 12-4 record, and denied the New England Patriots a chance to defend their title from Super Bowl XLIX by defeating them 20-18 in the AFC Championship Game. They joined the Patriots, Dallas Cowboys, and Pittsburgh Steelers as one of four teams that have made eight appearances in the Super Bowl.

  Predicted answer without a distractor: 15-1

- Distractor 1: 656 was the win/loss ratio in 2020 for the Michigan Vikings during their irregular season.

  Predicted answer with a distractor: 656.

- Distractor 2: The 2020 regular season win/loss ratio for the Michigan Vikings was 656.

  Predicted answer with a distractor: 656.

- Distractor 3: During their irregular session in 2020, the win/loss ration for the Michigan Vikings was 656.

  Predicted answer with a distractor: 656.

- Distractor 4: The Michigan Vikings had a 656 win to lose ratio in their 2020 off-season.

  Predicted answer with a distractor: 656.

b) Example 2

- Question: Which famous Indian practiced civil disobedience?

  Gold-label answer: Gandhi

- Context: Following the Peterloo massacre of 1819, poet Percy Shelley wrote the political poem The Mask of Anarchy later that year, that begins with the images of what he thought to be the unjust forms of authority of his time\u2014and then imagines the stirrings of a new form of social action. It is perhaps the first modern[vague] statement of the principle of nonviolent protest. A version was taken up by the author Henry David Thoreau in his essay Civil Disobedience, and later by Gandhi in his doctrine of Satyagraha. Gandhi's Satyagraha was partially influenced and inspired by Shelley's nonviolence in protest and political action. In particular, it is known that Gandhi would often quote Shelley's Masque of Anarchy to vast audiences during the campaign for a free India.

Predicted answer without a distractor: Gandhi.

- Distractor 1: The famous Pakistani practiced civil obedience.

  predicted answer with a distractor: Pakistani.

- Distractor 2: A famous Pakistani practiced civil obedience.

  Predicted answer with a distractor: Pakistani.

4) Conclusion

The analysis of general errors strongly suggests that the model is extremely likely to predict the wrong answer inferred from the given distractor when it is quite similar to the question sentence. By contrast, human readers would naturally exclude distractors however similar to the question if its crucial information is not identical. For example, In answering to the question "Which Indian practiced civil obedience?", a distractor that "A famous Pakistani practiced civil obedience" would not confuse a human reader. Unfortunately, existing NLP models may predict "Pakistani" as the answer because the rest of the words in the distractor are included in the question *verbatim*.

In another example, due to a high similarity between the question "Which Middle Eastern nation in particular views Genghis Khan as a contemptible perpetrator of genocide? " and the distractor "The East Western nation in particular views Kublai Shah as a estimable perpetrator of genocide of Chicago.", the model answers "East Western nation." However, despite another distractor "The western nations in particular view Kublai Shah as a estimable perpetrator of genocide in Chicago," the model still yielded the correct answer "Iran." It could be because "Iran" is taught to be very contrasting to the "West" from its training text sources.

Considering that the model correctly answers the question despite the presence of distractors, as shown in Specific errors 2, the model seems to compare the word similarities between the distractor and the question. Nonetheless, in many cases the model fails to capture a crucial difference among the given words (e.g., West v. East).

III. Stacking Ensemble to Reduce Dataset Artifacts

A. *Decision to employ stacking ensemble*

I set high expectations on this project myself because NLP is what has drawn me to the computer science field. QA is the most interesting research agenda, and I believe a meaningful breakthrough in QA performance will bring us into a step closer to a next-level general artificial intelligence. I meticulously read the papers listed in the final project briefing that are related to QA. Among them, Clark et al. (2019) argued that "State-of-the-art models often make use of *superficial patterns* in the data that do not generalize well to out-of-domain or *adversarial* settings. For example, textual entailment models often learn that *particular key words* imply entailment, irrespective of context (italics added)." My analysis of the performance of ELECTRA, where it is trained on SQuAD and then tested against Adversarial SQuAD, aligns with their assessment.

Clark et al. (2019) thus proposed an ensemble method to reduce biases. However, what they used to achieve bias reduction is rather unique, which does not belong into one of the three typical ensemble techniques: bagging, boosting, and stacking. **Their solution inspired me to use an ensemble method as well**. After a brief review of the machine learning ensemble methods, I concluded that stacking could improve the performance of ELECTRA by setting multiple trained models as base models (Level 0) and introducing a meta-learner (Level 1) that uses the results (logits) of base models as inputs and learns a new pattern.

Even though I was able to find many sources on how to implement stacking ensemble methods with conventional machine learning methods like regression and support vector machines as base models, there has been little guide or research that applies stacking to neural networks, let alone, the Hugging Face platform.

B. *Overview of Stacking Ensemble Process*

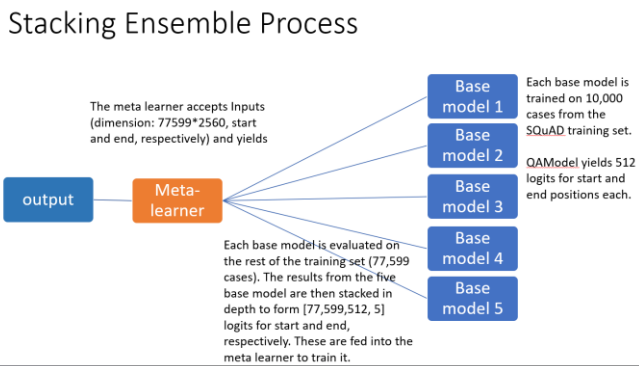

Stacking Ensemble Process
The meta learner accepts Inputs (dimension: 77599*2560, start and end, respectively) and yields
output
Meta-learner
Base model 1
Base model 2
Base model 3
Base model 4
Base model 5
Each base model is trained on 10,000 cases from the SQuAD training set.
QAModel yields 512 logits for start and end positions each.
Each base model is evaluated on the rest of the training set (77,599 cases). The results from the five base model are then stacked in depth to form [77,599,512, 5] logits for start and end, respectively. These are fed into the meta learner to train it.

FIGURE I. AN OVERVIEW OF STACKING ENSEMBLE METHOD

Stacking ensemble includes two phases of training. The first phase is training base models, and these base models could be a set of different models. For example, they could be a portfolio of encoder-only models like BERT, RoBERTa, ELECTRA, etc. However, using different models as bases can be time-consuming because each base needs to be trained separately. Instead, I trained the same ELECTRA-small models five times and created separate base models. One difference from the conventional training process is that in this training process, you need to divide the original training dataset into two: one for base models, and the other for the meta-learner. I used 10,000 of 87,599 training examples (or data points) in SQuAD to train each base model.

Then, these base models are evaluated against the rest of the training set, 77,599 examples. The second phase of training uses these evaluation results of the base model as inputs to train the meta learner. The meta learner uses predictions of five models as inputs and learns a new pattern, which will contribute to improved performance compared to the base model. After the training of the meta learner is completed, the original evaluation dataset is used to test the meta learner's performance.

The meta-learner can be as simple as a voting mechanism, logistic or linear regression, or a separate neural network.

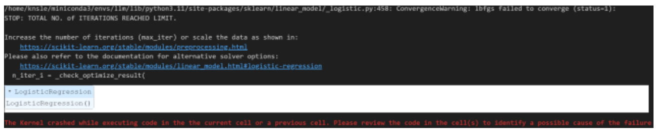
C. *Reverse-engineering of the Hugging Face Library: a farily daunting task for QA models*

1) *Issues regarding the project*

The Hugging Face libraries provide great resources and convenience to use modern large language model (LLM) architectures. With the import of its module (transformers), Python can directly import learned parameters (or checkpoints) from the Hugging Face hub as well as model architecture. In the case of QA, Hugging Face‘s AutoModelForQuestionAnswering will load the architecture with appropriate heads and checkpoints. In addition, Hugging Face offers auto-tokenizers and trainers which can be done in even one line of code.

Unfortunately, such ease of use could become a double-edged sword if you are not familiar with the inner workings of Hugging Face. If you want to initiate a project that needs customization, you have to reverse-engineer Hugging Face’s interfaces which are ready-made for most customers’ needs. Especially if you are working on a QA project, it becomes more complicated.

First, I had to code to train the model manually. QA models output two logits for start and end positions. That is, each model’s output is an array of [2, 77,599, 512]. Two refer to start and end positions, 77,599 is the total number of training data points minus 10,000, and the model predicts logits of 512 tokens. Since there are five models, the final output will be stacked up to be a 3D array of [2, 77,599, 512, 5]. Next, you need to set up a new training set for the meta learner. The meta learner accepts each model’s output equally, so the shape of the input array should be transformed into [2, 77,599, 2560].



```
/home/knsle/miniconda3/envs/llm/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

▾ LogisticRegression
LogisticRegression()

The Kernel crashed while executing code in the the current cell or a previous cell. Please review the code in the cell(s) to identify a possible cause of the failure
```

FIGURE II. SCREENSHOT OF FAILED CONVERGENCE IN LOGISTIC REGRESSION

2) *Logistic regression as meta learner*

As the first candidate for the meta learner, I set up two logistic regression models for start and end positions. Each model uses 2,560 inputs from 77,599 examples and predicts 512 logits. The gold label is the ground-truth answers from the training data set. Unfortunately, this did now work well. The model failed to converge after the total iteration reached the limit. It seems that the output dimensions are too large.

3) *Neural Network as meta learner*

As the second candidate, I formed a feedforward network that accepts 2,560 inputs and finally outputs two numbers (starting and ending positions).

D. *Result*

1) *Summary Statistics*

Validation with Adversarial SQuAD	**Summary Statistics**	
	Match Rate	***F1***
Meta learner	40.12	44.35

TABLE II. VALIDATION RESULTS FOR THE META LEARNER FROM THE STACKING ENSEMBLE

The validation result of the meta learner is shown in Table II. The result is not so satisfactory: The match rate is only 40.12, and the F1 score is 44.35. This is even lower than ELECTRA-small’s performance on Adversarial SQuAD. Unfortunately, the stacking ensemble did not work as intended.

2) *Analysis of Results*

A rather disappointing result may have been caused by several factors. First, this project used five base models from the same architecture ELECTRA-small. And these models are trained on the same part of SQuAD’s whole training data (10,000 out of 87,599 examples). While this approach may not be wrong, it could have produced base models with little-to-no difference from one another. It would have been better if I had adopted different encoder-only models like BERT, RoBERTa as base models.

Next, due to the time constraints, I could not test various hyper-parameters of the meta learner. I could have changed the number of layers and the number of inputs and outputs in the hidden layers. I used 1,000 epochs and the loss kept decreasing. So, if I had trained the meta learner more or used a more effective design, the meta learner could have performed better.

Finally, to complete this project, I more deeply researched Hugging Face interface and QA tasks. Thanks to more research, I realized that there is a significant difference between the training set and the validation set. As for SQuAD’s training data, there is *only one* answer per example, which makes sense given the nature of training. However, for validation purposes, it does not matter where the correct answer’s indices are. As long as the model identifies the correct answer that might appear multiple times in the context, the model’s performance is satisfactory, so the SQuAD validation set lists multiple starting and ending indices of characters.

Related to this difference between training and validation examples, the dilemma of the stacking ensemble method originates from the second phase of the process. The base models are evaluated on the training data not used for training those base models (77,599 examples). So, even if the base models predict the correct answer from different indices, they will be marked as wrong, which leads to false negatives in training the meta learner. Since most of the original training data is used for validating the base models and subsequently training the meta learner, this false negatives must have taught the meta learner in the wrong direction.

IV. ALTERNATIVE APPROACH: TRAINING ON SQUAD V2

A. *Train ELECTRA on SQuAD V2*

Since I did not want to submit a report on an unsuccessful project, I decided to make a quick fix on the project issue. SQuAD was introduced in 2016, and SQuAD V2 in 2018 includes questions that do not have an answer in the context [7]. If SQuAD V2 is merely an extension of the original SQuAD with the same pattern of QA sets, it would aggravate the model’s reliance on dataset artifacts or superficial similarities. By contrast, QA sets that do not have answers could weaken that trend and train the model in the right direction. Thus, I suspected that training on SQuAD V2 could weaken the model’s reliance on superficial relationships between the question and the context and improve the performance when evaluated against Adversarial SQuAD.

Validation with Adversarial SQuAD	Summary Statistics	
	Match Rate	***F1***
ELECTRA trained on SQuAD	50.28	57.57
ELECTRA trained on SQuAD V2	55.96	63.13

TABLE III. COMPARISON OF VALIDATION RESULTS OF ELECTRA-SMALL FINE-TUNED ON SQUAD OR SQUAD V2

B. *Results*

Table III shows the results of training ELECTRA-small on SQuAD V2. Its performance on Adversarial SQuAD improved to some degree. Its match rate rose by 5.7 percent and the F1 score also increased by about 6 points. So, a quick fix on the project was somewhat successful.

V. Final Thoughts and Conclusion

This project provided me with immense opportunities to learn more about NLP, Hugging Face, and QA tasks. I spent so much time studying and reverse-engineering Hugging Face libraries and classes related to QA tasks. I learned a lot from such an experience. I am planning to continue with this project, using multiple base models and trying different designs for the base models.

I would like to add that the instructions in the final project could mislead. It suggests only minor coding is needed for this project, and that much effort should be directed to designing the improvements. In fact, since we have not learned anything about Hugging Face and QA tasks, it requires a lot of coding and researching Hugging Face in general before getting our hands on the project.

Acknowledgments

Starter codes for this project have been provided in a graduate computer science course CS388 Natural Language Processing at University of Texas, Austin. The author has made significant revisions and additions to those codes to fulfill the objective of this project.

References

[1] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4069–4082, Hong Kong, China. Association for Computational Linguistics.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[3] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Advances in Neural Information Processing Systems. pages 1693–1701.

[4] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. arXiv preprint arXiv:1511.02301

[5] 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, September. Association for Computational Linguistics

[6] 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, November. Association for Computational Linguistics

[7] 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.

[8] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. ArXiv:1910.03771.