Reproducibility Report for CSE 481N

Griffin Golias, Alex Dundarov, Charles Immendorf, Eric Yeh
Team 4 - Paul G. Allen's Card
{goliagri, alexd02, chazi, yehe}@uw.edu

May 14, 2023

Project Report

Paper: Logical Fallacy Detection [1]

1 Introduction

Jin et al. [1] creates 2 new datasets for a logical fallacy detection task. Given a short statement, a model is tasked with predicting which of 13 logical fallacy classes is present. The authors introduce a general Logic dataset and a thematic Logic-climate dataset for evaluating on logical fallacy detection. An NLI model is used with the statement as the premise and a hypothesis that this is an example of a given logical fallacy. The authors propose a "struct-aware" modification to a pretrained Electra model which applies a masking to the premise to emphasize its logical form and uses a hypothesis asking if the logical form matches a fixed base logical form for each fallacy. They report results that the struct aware version of Electra outperforms the original's accuracy score by about 12% on Logic, 10% trained on Logic and evaluated on Logic Climate, and 44% on Logic Climate.

2 Scope of Reproducibility

This paper introduces a structure-aware model called Electra-StructAware and claims that this model classifies logical fallacy arguments better than other models. The reasoning is that the structure of an argument is more likely to determine the logical fallacy type of a fallacious argument than the content words. Therefore, Electra-StructAware masks out the content words in the sentence so it is stripped down to the basic structure of the sentence.

To corroborate the claim, the paper ran experiments using the Electra-StructAware model and compared the results against other zero-shot and fine-tuned models and outperforms the best one, giving an F1 score of 58.77%. Since the other models are no longer available, we have focused solely on reproducing the results of the Electra scores to determine if the claim holds.

We seek to reproduce the author's original f_1, precision, recall, and accuracy scores for the models

- Electra
- Electra Struct-aware

Evaluated on

- Logic with model Finetuned on Logic
- LogicClimate with model finetuned on Logic
- LogicClimate with model finetuned on Logic and LogicClimate

3 Methodology

3.1 Model Descriptions

The Electra-StructAware model produces in a structure-aware premise and a structure-aware hypothesis and passes those into a NLI model that will determine if the structure-aware premise matches the structure-aware hypothesis.

The structure-aware premise consists of the original sentence whose fallacy type that is to be predicted. The sentence is modified using coreference resolution to mask out the similar content words. After masking, the sentence is simplified to the basic structural elements that might make it easier to determine the fallacy type.

The structure-aware hypothesis consists of a statement predicting the logical fallacy type of the structure-aware premise. The name of the predicted label in the hypothesis statement is replaced by the logical form of the statement. For example, instead of passing in "This is an example of deductive fallacies", the structure-aware model uses "The example matches the logical following form: 'If [MSK1], then [MSK2]' leads to 'If [MSK2], then [MSK1]'"

3.2 Datasets

We utilize 2 different datasets for training and evaluation: LOGIC and LOGICCLIMATE. The links to the articles from these datasets can be found on the original Github project repository.

LOGIC is a general dataset consisting of examples of logical fallacy types pulled from various educational sources, such as Quizziz, study.com, and ProProfs. This dataset consists of 2449 logical fallacy instances split among the 13 logical fallacy types.

LOGICCLIMATE is a challenge dataset with climate change claims that have been manually extracted from climate change news articles on the Climate Feedback website by native English speakers. The annotators selected specific text spans within the article and assigned a logical fallacy type label to each.

3.3 Implementation

We have utilized the code that is provided on the Github repo. For the main training and evaluating, we ran the scripts found in codes_for_models/logicedu.py. We also generated additional wrapper scripts to run logicedu.py at various thresholds to determine the impact on scoring metrics. These scripts can be found in our forked version of the repo, located at https://github.com/CharlesAtUW/logical-fallacy/

3.4 Experimental Setup

We ran the experiments on the NLP machines, which have 2 GPUs. We also set up a virtual Python environment with all the dependencies needed to run the scripts.

The following command was run to evaluate the Electra model:

```
$ python logicedu.py  -t "google/electra-large-discriminator"
    -m "../../saved_models/electra-logic/"
    -ts 1 -ds 1 -nt T -mp base
```

The following command was run to train and evaluate the Electra-StructAware model:

```
$ python logicedu.py -t "google/electra-large-discriminator"
    -m "howey/electra-base-mnli"
    -ts 1 -ds 1 -nt F -mp masked-logical-form -w 12
    -s ../../saved_models/new-struct-aware-model/
```

3.5 Precision vs. Recall

We also test how the model's precision and recall change when we change the threshold value (which can be from 0 to 1) for which an input is considered a fallacy or not. To see how we change the threshold values, we first look at how the model makes predictions originally. For each input sample into the model, and for each of the 13 fallacies, the model will classify if the sample has the fallacy ("entailment"), doesn't ("contradiction"), or neutral ("neutral"), by outputting a number for each of these classes and classifying the class with the highest number. (However, none of

the examples, at least in the test set, have the ”neutral” label, and none of the models tested ever classified any input as ”neutral”. So effectively, binary classifications were made.) To make a situation where we can change the threshold, we subtract the ”contradiction” number from the ”entailment” number, then apply the sigmoid function to it. Note that a threshold value of 0.5 corresponds to what the model does normally. Lower threshold values mean that a larger range of sigmoid outputs get classified as ”entailment”.

4 Results

We were successfully able to reproduce the results of the Electra model, getting fairly similar results to the original paper. However, we have found that the results of rerunning the Electra-StructAware model do not reflect the results reported in the original paper, even after retraining the model from scratch. We will need to explore further as to why our results are not matching and how we can adjust our configuration to reproduce successfully.

4.1 Electra

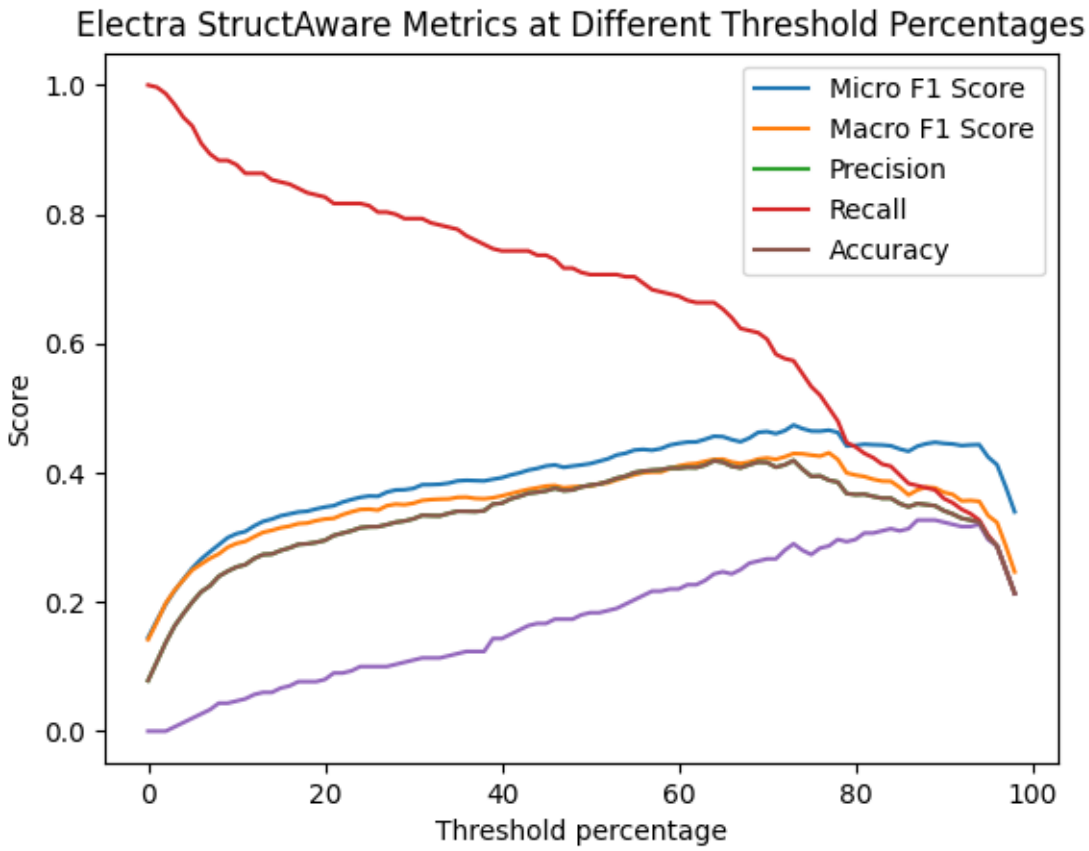
	Reproduced	Original
F_1	50.58	53.31
P	51.23	51.59
R	80.33	72.33
Acc	33.33	35.66

4.2 Electra-StructAware

	Reproduced	Original
F_1	47.95	58.77
P	47.11	55.25
R	74.00	63.67
Acc	29.33	47.67

4.3 Electra-StructAware

Because our model was reaching the desired results, we decided to test out the impact of different thresholds on the accuracy scores to see if changing it would get us closer to the reproduced results. The scores from adjusting the thresholds are reflected in the graph below.



Unfortunately, we were not able to get our accuracy score much higher than 0.4, in comparison to the desired result of over 0.5. We will continue conducting other experiments to see if we can make further progress in reproduction while we wait for the authors to respond.

5 Discussion

Unfortunately, although we have been able to rerun the experiments in the original paper, we have had little luck in reproducing the paper's results. While Jin et al's findings demonstrate an increase in model accuracy after introducing structure awareness, our results display a decrease, even after finetuning the classification threshold.

5.1 What was Easy

The paper's experiments were easy to run because the codebase was provided for us. Additionally, we got some guidance from the authors and other reproducers on what commands to use to run the code. Most of the resources were accessible for our team, including GPU machines to run the experiments more quickly and the datasets which were provided in a Google Drive linked on the main repository.

5.2 What was Difficult

Although a repository was provided for us, we struggled with having poor documentation throughout the codebase, making it difficult to figure out how the components of the project worked together. This made it difficult to trace down the areas of the code that may explain why our results were different despite having the same experimental setup.

Communication with Original Authors

We have contacted the lead author of the paper, Zhijing Jin, and the main contributor to the Github repo, Abhinav Lalwani. We have learned that the reproducibility issue on the Github repo has not yet been resolved, and they encourage us to work towards getting it working properly.

Questions to Ask:

1. Explanation of the results, and why some of the accuracies are 0s?
2. More context on how the model can be trained and evaluated, which scripts to run (including what commands to pass in)
3. What further work can be done to build on the results from the paper?

References

[1] Z. Jin, A. Lalwani, T. Vaidhya, X. Shen, Y. Ding, Z. Lyu, M. Sachan, R. Mihalcea, and B. Schoelkopf. Logical fallacy detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7180–7198, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. 1