Project 3: Dataset Artifacts

Academic Honesty: Please see the course syllabus for information about collaboration in this course. While you may discuss the assignment with other students, **all work you submit must be your own!**

Goals The primary goal with this assignment is to give you understanding of how Transformer models learn spurious correlations on datasets for natural language inference (NLI), then give you a sense of some possible techniques to improve these spurious correlations. Compared to past projects, the emphasis is much more heavily on looking at and analyzing data compared to building models.

General Description Pre-trained models can often achieve high performance on benchmark datasets, but are they really "solving" the tasks these datasets encapsulate? Sometimes a model can work extremely well even when presented with a modified version of the input where it should not be possible to predict the right answer, like hypothesis-only baselines in NLI (Poliak et al., 2018; Gururangan et al., 2018), which calls into question what the model is even learning. Sometimes it is possible to find or construct examples very similar to those in the training data where the model achieves surprisingly low performance. These include "contrast examples" (Gardner et al., 2020; Li et al., 2020) which are produced by modifying real examples in a small way, as well as adversarial examples (Jia and Liang, 2017) and checklist examples (Ribeiro et al., 2020).

These observations all stem from the fact that a model may achieve high performance on a dataset by learning spurious correlations, also called **dataset artifacts**. The model is then expected to fail in settings where these artifacts are not present, which may include real-world testbeds of interest.

Your task is to investigate the performance of a pre-trained model on the Stanford NLI (SNLI) dataset (Bowman et al., 2015). **You will analyze the model's performance and shortcomings, try to improve it, and describe whether what you did fixed things.**

Dataset and Code

Installation instructions Please follow the instructions in the GitHub repository here: `https://github.com/gregdurrett/fp-dataset-artifacts`. **Note that this repository also supports SQuAD, which you will not be using. You only need to pay attention to the NLI/SNLI commands.**

Starter code In the repository above, you'll find `run.py`, a script which implements basic model training and evaluation using the Hugging Face `transformers` library. For information on the arguments to `run.py` and hints on how to extend its behavior, see the comments in the source and the repository's `README`.

Hugging Face The skeleton code is heavily based on Hugging Face `transformers`, which is an open-source library providing implementations of pre-trained deep learning models for a variety of (mainly NLP) tasks. If you want to get more familiar with `transformers`, you can check out the examples in their GitHub repository.

Computational resources Even with the ELECTRA-small model, training with CPU only on a large dataset like SNLI can be time-consuming. Please see the section on **Compute** at the end of the project spec for some options.

Part 1: Analyzing Artifacts (50 points)

You should start by doing some analysis of the ELECTRA-small (Clark et al., 2020) model on the SNLI dataset. ELECTRA has the same architecture as BERT with an improved training method, and the small model is computationally easier to run than larger models. **Your goal in this part is to identify and attempt to quantify a spurious correlation that a fine-tuned ELECTRA model picks up on. Although you can try to study something unexplored if you want, we recommend you simply reproduce and extend the analysis from prior work using one of the techniques below.**

There are a few different analyses you can do. **You should pick one.**

- **(changing model)** Use model ablations like hypothesis-only models for NLI (Gururangan et al., 2018; Poliak et al., 2018). Other simplifications are possible too.
- **(changing data)** Use contrast sets (Gardner et al., 2020); these "minimal pair" examples can reveal features of the models' behavior by altering labels without altering other correlations. A contrast set was constructed for SNLI by Li et al. (2020).
- **(statistical test)** Use the "competency problems" framework: find spurious n-gram correlations with answers (Gardner et al., 2021).

Loading your own data See `https://huggingface.co/docs/datasets/v1.1.3/loading_datasets.html#json-files` for instructions on how to load datasets from json. You will want to prepare a dataset with the form described in the comment of `run.py`.

Writing up Part 1 You should train an ELECTRA-small model, then analyze it using one of these frameworks, reporting necessary numbers and describing what they tell us about spurious correlations. Specifically, your report should address the following points:

1. **What artifact, bias, or spurious correlation are you testing for?** Describe it as precisely as possible.
2. **What experiments did you set up in order to test this?** Describe the models and datasets you used. Note: you do not need to give significant details about standard models like ELECTRA or standard datasets like SNLI (e.g., number of examples) except insofar as these details are important to your analysis (e.g., the length distribution of hypotheses in the dev set could be meaningful if you're investigating correlations with length).
3. **Do you see evidence for presence of that artifact?** Report evidence in graphs and tables. You should take care to compare anything you run to appropriate baselines to develop your argument around it.

Part 2: Fixing It (50 points)

Pick a method to try and improve the issues you identified in Part 1. Some options are listed below, following the discussion in lecture:

- Focusing learning on hard subsets of data or data where the gold label distribution is ambiguous. Dataset cartography (Swayamdipta et al., 2020) is a good framework for identifying such examples, but there are many options (Yaghoobzadeh et al., 2021; Nie et al., 2020; Meissner et al., 2021).

- Ensemble-based debiasing using artifact experts: train a weak or partial model to learn the correlations, then train your model to learn the *residual* of that model (He et al., 2019) or otherwise remove it from output distribution (Clark et al., 2019; Zhou and Bansal, 2020; Utama et al., 2020; Sanh et al., 2021).
- Training on adversarial data, including using challenge sets directly or adversarial data augmentation (Liu et al., 2019; Zhou and Bansal, 2020; Morris et al., 2020)

You are allowed to use open-source GitHub repositories associated with these papers if you want. However, we do not guarantee that any of these work or are easy to get working; you might find that implementing the ideas yourself in your own code framework is actually easier, as most of them are not too challenging to code up.

Writing up Part 2 Evaluate your fix. How effective is it? Did you manage to address the errors you were targeting? How broad was this fix; did it address other related errors or issues as well? Did it make overall dataset performance go up or down? **It will be very hard to get overall much stronger performance on an in-domain test set**, but well-implemented projects should be able to show some improvement either on a subset of examples or on generalization to a challenging setting.

Report both results in tables as well as any other pertinent analysis to back things up, such as results in your Part 1 setting or results on subsets of the data. If you tried multiple things, analyze the contribution from each one with an *ablation study*. These should be *minimal* changes to the same system; try running things with just one aspect different in order to assess how important that aspect is. If you just had one change, then make sure that you clearly report results from your baseline and your improved system.

Compute

The most important thing is to **only run large-scale experiments when needed.** Debug things on small amounts of data. Depending on what dataset you have and what resources you're using, you may need to let your initial model train for a few hours to get a usable initial result, and your final experiments may be similarly time-consuming, but ideally much of what you do in the middle won't need to involve waiting hours for models to train. Note: you do not need to stick with the default parameters we give you, and training for less time or on less data than in SNLI may still give reasonable performance to do your investigation.

Using Checkpoints The Hugging Face trainer checkpoints the models periodically. As a result, you can start a long training run, leave it going for a while, and evaluate how much of that time was actually needed to get good performance. If it worked well after only two hours, you'll know that for the future. Ideally, you can then do further experimentation more quickly or even start future runs from checkpoints.

GCP Google Cloud Platform offers free credits upon signing up for a new account, which are more than sufficient to run some large-scale experiments for the course. The course staff are able to provide limited support on how to use GCP, but you'll mostly have to figure this out yourself.

Google Colab Google Colab is another resource for exploring training on GPUs. You can see how to get started with HuggingFace on Colab here: `https://github.com/huggingface/transformers/tree/master/notebooks`

You can also look into Colab Pro, which is a paid membership but gives you access to more resources. As of February 2023, it's $9.99 per month.

Writeup and Code Submission

Writeup

You should submit a 2-3-page writeup describing what you did and your results in both Part 1 and Part 2. It should follow the style of the previous writeups, but be a bit more in-depth due to the more open-ended nature of this project and the lack of an autograder.

The project will be graded solely on the basis of the writeup. We will grade roughly equally across the two parts, and within each part we will be evaluating:

- **Scope (10)**: Was a sufficient amount of work done according to what's laid out in the project specification?
- **Implementation, Results, Analysis (30 points)**: Is the implementation reasonable? Were appropriate results reported? Was the analysis of the results correct, and were sound conclusions drawn based on the evidence? Whether the results are positive or negative, try to motivate them by providing examples and analysis. If things worked, what types of errors are reduced? If things didn't work, why might that be?
- **Clarity/Writing (10 points)**: Was the report clearly written and structured to convey the ideas?

Code Submission

There is no autograder for this assignment. You will upload your code for Part 1 and Part 2 on Canvas. **The code upload is purely documentary – we will not attempt to run it. Do not upload any data, model outputs, or model checkpoints. YOUR FINAL UPLOAD SHOULD BE LESS THAN 500 KB.**

References

[Bowman et al.2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.

[Clark et al.2019] 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, November. Association for Computational Linguistics.

[Clark et al.2020] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Gardner et al.2020] Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hanna Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets.

[Gardner et al.2021] Matt Gardner, William Merrill, Jesse Dodge, Matthew E Peters, Alexis Ross, Sameer Singh, and Noah Smith. 2021. Competency problems: On finding and removing artifacts in language data. *arXiv preprint arXiv:2104.08646*.

[Gururangan et al.2018] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana, June. Association for Computational Linguistics.

[He et al.2019] He He, Sheng Zha, and Haohan Wang. 2019. Unlearn dataset bias in natural language inference by fitting the residual. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 132–142, Hong Kong, China, November. Association for Computational Linguistics.

[Jia and Liang2017] 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.

[Li et al.2020] Chuanrong Li, Lin Shengshuo, Zeyu Liu, Xinyi Wu, Xuhui Zhou, and Shane Steinert-Threlkeld. 2020. Linguistically-informed transformations (LIT): A method for automatically generating contrast sets. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 126–135, Online, November. Association for Computational Linguistics.

[Liu et al.2019] 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, June. Association for Computational Linguistics.

[Meissner et al.2021] Johannes Mario Meissner, Napat Thumwanit, Saku Sugawara, and Akiko Aizawa. 2021. Embracing ambiguity: Shifting the training target of NLI models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 862–869, Online, August. Association for Computational Linguistics.

[Morris et al.2020] John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp.

[Nie et al.2020] Yixin Nie, Xiang Zhou, and Mohit Bansal. 2020. What can we learn from collective human opinions on natural language inference data? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9131–9143, Online, November. Association for Computational Linguistics.

[Poliak et al.2018] Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana, June. Association for Computational Linguistics.

[Ribeiro et al.2020] 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, July. Association for Computational Linguistics.

[Sanh et al.2021] Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M Rush. 2021. Learning from others' mistakes: Avoiding dataset biases without modeling them. In *International Conference on Learning Representations*.

[Swayamdipta et al.2020] Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online, November. Association for Computational Linguistics.

[Utama et al.2020] Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Towards debiasing NLU models from unknown biases. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7597–7610, Online, November. Association for Computational Linguistics.

[Yaghoobzadeh et al.2021] Yadollah Yaghoobzadeh, Soroush Mehri, Remi Tachet des Combes, T. J. Hazen, and Alessandro Sordoni. 2021. Increasing robustness to spurious correlations using forgettable examples. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3319–3332, Online, April. Association for Computational Linguistics.

[Zhou and Bansal2020] Xiang Zhou and Mohit Bansal. 2020. Towards robustifying NLI models against lexical dataset biases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8759–8771, Online, July. Association for Computational Linguistics.