CS378 Natural Language Processing: Final Project

Check-In Due Date (for all projects): Friday, November 18 at 11:59pm CST (no slip days)
Final Report Due Date: Friday, December 9 at 11:59pm CST (no slip days)

Collaboration You are free to work on this project in teams of two (encouraged) or individually. Individual projects can be less ambitious but should not be less complete: a half-implemented system does not make a good project outcome. All partners should contribute equally to the submission, and all partners will receive the same grade for it. You are also free to discuss your project with others in the course, though only the people on your team should contribute to the actual implementation/experimentation involved. Any external resources used must be clearly cited.

If you're pursuing an independent project, you **may** collaborate with someone from outside the course. You also **may** use an independent project as a final project for another course as well, **provided you have the approval of the other professor**.

Assignment

You have two options for your final project:

1. An investigation into dataset artifacts
2. An independent project of your choosing

1 Analyzing and Mitigating Dataset Artifacts

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), 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) 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 a task of your choice. We recommend one of the following datasets from either natural language inference (NLI) or question answering (QA):

1. The Stanford NLI dataset (Bowman et al., 2015)
2. MultiNLI (Williams et al., 2018)
3. SQuAD (Rajpurkar et al., 2016)
4. HotpotQA (Yang et al., 2018)

You will analyze the model's performance and shortcomings, try to improve it, and describe whether what you did fixed things.

1.1 Part 1: Analysis

You should start by training a model on your selected dataset and doing some analysis of it. **We provide starter code for this. We recommend using the ELECTRA-small (Clark et al., 2020) model**; ELECTRA has the same architecture as BERT with an improved training method, and the small model is computationally easier to run than larger models. However, you are free to use any model you'd like. See Section 1.3 for details on the starter code.

There are many ways to conduct this analysis:

- **(changing data)** Use contrast sets (Gardner et al., 2020), either ones that have already been constructed or a small set of examples that you hand-design and annotate
- **(changing data)** Use checklist sets (Ribeiro et al., 2020)
- **(changing data)** Use adversarial challenge sets (Jia and Liang, 2017; Wallace et al., 2019; Bartolo et al., 2020; Glockner et al., 2018; McCoy et al., 2019)
- **(changing model)** Use model ablations (hypothesis-only NLI, a sentence-factored model for multi-hop question answering, a question/passage only model for QA) (Poliak et al., 2018; Chen and Durrett, 2019; Kaushik and Lipton, 2018)
- **(statistical test)** Use the "competency problems" framework: find spurious n-gram correlations with answers (Gardner et al., 2021)

We recommend you briefly skim the abstracts of papers for 2-3 of these approaches to see what makes sense. You won't have time to investigate all of these, so an important part of the process is prioritizing what you want to try and assessing whether it makes sense for the dataset you're exploring.

Writeup In your writeup, you should (1) give some examples of both **specific errors/behavior** from the model as well as analysis or discussion of **the general class of mistakes the model makes**. For characterizing the general class, try to come up with rules that can identify a set of challenging examples (e.g., "examples containing *not*") and try to visualize in charts, graphs, or tables what you believe to be relevant statistics about the data. **This part of the report should probably be at least a page long.**

1.2 Part 2: Fixing it

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

- 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)
- Contrastive training (Dua et al., 2021)

You are allowed to use open-source GitHub repositories associated with these papers. If you do choose to build on a repository used in other work, you should consider running on another dataset, trying some twists on their methodology, analyzing across different dimensions, or other modifications. If you do not choose to follow an existing repository, **it's fine to structure your report as a reproduction effort.** In this case, you might describe how your results differ from theirs.

Writeup 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 we expect that well-implemented projects should be able to show some improvement either on a subset of examples or on generalization to a challenging setting.

You don't need to answer all these questions, but for what you try to answer, you should both compute accuracy numbers and show deeper analysis and visualization to back that up. For example, if your change made the model better on challenging NLI examples, you could try to quantify that on one or more slices of the data, give examples of predictions that are fixed, or even use model interpretation techniques to try to support claims about how your improved model is doing its "reasoning."

The analysis of your results is critical. You should think about your evaluation plan **before** diving into Part 2 and have a plan for what kinds of improvements or changes you expect to be able to show. If you feel like your change will have a very limited impact and not show up on any of the criteria you evaluate for, you may want to try something different. Alternatively, maybe your change fixes a small but important class of examples; it might improve performance on the hardest 10% of examples in the dataset. If you can show this, it's a fine outcome!

1.3 Getting Started

Installation instructions Please follow the instructions in the GitHub repository here: `https://github.com/gregdurrett/fp-dataset-artifacts`

Starter code In the repository above, you'll find `run.py`, a script which implements basic model training and evaluation using the HuggingFace `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`.

Note that you do not have to use the provided training script; you can also fine-tune the model with a training loop like the one from previous assignments.[1]

HuggingFace The skeleton code is heavily based on HuggingFace `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 or SQuAD can be time-consuming. Please see the section on **compute and feasibility** at the end of the project spec for some options.

[1] `https://huggingface.co/transformers/custom_datasets.html#fine-tuning-with-native-pytorch-tensorfl`

1.4 Example

Consider following up on the Dataset Cartography paper (Swayamdipta et al., 2020). If you use their repository rather than reimplementing the technique yourself, you should explore using the technique on a different dataset than the one they considered (e.g., consider applying it to the SQuAD dataset). Here are some further questions you might ask:

1. By using this technique, you can split the dataset into three subsets: easy-to-learn, hard-to-learn, and ambiguous; do the examples in each subset share something in common? What makes the examples hard to learn or easy to learn? What is the role each subset plays during training?

2. For the hard-to-learn and ambiguous examples, is there a way to make learning "pay more attention" to them? You can consider approaches beyond what they explore in their work, including data augmentation or soft reweighting of the dataset.

1.5 Scope

The "fix" you try does not strictly need to work in order to have a successful project. However, if it doesn't work, you should have a plan for how to analyze it and be able to dig into the data more. Just saying "I tried X and wrote 100 lines of code but it still crashes" is not a good project outcome. Try to make sure you're on track to have some preliminary results or analysis supporting what you're trying to do a week or two out from the deadline. From there, make sure that even if things don't work, you can still argue (a) that you've correctly implemented what you set out to implement; (b) you can analyze the results to understand why things went wrong.

Note that you may not end up writing all that much code for this project. There may be great projects that really only involve modifying or adding 20 lines code on top of the implementation we give you. Much more of the work lies in (a) studying the data; (b) understanding the modifications you're making; (c) analyzing your fix.

One person vs. two person projects If you're working in a group of two, we expect the work to scale up appropriately.

As a single-person project: You probably want to stick closer to the settings discussed in prior work, or try a more straightforward method to modify the model training. You are still expected to carry out the same basic analyses, but they do not need to be as detailed.

As a two-person team: You might want to try a more sophisticated modification to the basic modeling framework, or you could even try two different modifications to tackle the same problem (or two variants of the same modification). Your analyses are expected to be more in-depth than for a one-person project.

Compute and Feasibility See the end of the project spec for more discussion about this.

2 Your own project

You also have the option to pursue your own independently-chosen final project. Your proposal in this case should, as precisely as possible, describe the problem you want to solve or understand and the work you propose to do to solve it. Make sure you describe any datasets you plan to use and demonstrate that what you're proposing is feasible. Working on adjacent areas of machine learning (e.g., computer vision or

robotics) is okay as long as there is a reasonable connection to concepts from this course. Ask the course staff if you're unsure.

Scope The expected scope should be similar to the QA project. In particular, you should either have a nontrivial piece of implementation (putting together existing pieces can be nontrivial) or analysis. The project should also engage with the course concepts. One example of an **inappropriate** project is scaling an existing machine translation to run on a cloud framework: this may have some challenges and is probably a sufficient amount of work, but mostly involves engineering and concepts not related to linguistics, machine learning, or algorithms as discussed in class.

Proposal For independent projects only, there is a proposal due before the official start of the final project period (after Assignment 5). This proposal should be **1/2 to 1 page**. In it, you should describe the problem you plan to solve, what data you plan to use, any existing systems you plan to start from, and what your estimate of the computational requirements will be. We will evaluate these proposals primarily to see whether your idea is appropriate in scope and whether the project is feasible: will you run into issues with lack of data or proprietary data, and will the project's computational requirements be realistic given the resources you have available.

3 Deliverables and Grading

Check-In (10 points) You should turn in a check-in (at least a couple paragraphs, no more than 1 page) on check-in due date (a bit less than halfway through). This check-in should outline what you've looked into so far and what your plan is for the remainder of the project (e.g., "I/we want to investigate X, we got the basic system running and looked through 10 data examples to confirm that our idea might work"). The course staff will then provide feedback and guidance on the direction to maximize the project's change of succeeding. The check-in is worth 10 points and is graded based on whether you provided a good assessment of your progress so far, *not* the progress itself.

Code You should submit any code you wrote on the project due date, but this is **for documentary purposes only**; we will not be attempting to verify your results by running the code. Please do not include large data files or external resources you used that might be needed to execute it.

Final Report The primary deliverable is a paper written in the style of an ACL[2]/NeurIPS/etc. conference submission.[3] It should begin with an abstract and introduction, clearly describe the proposed idea or exploration, present technical details, give results, compare to baselines, provide analysis and discussion of the results, and cite any sources you used.

This paper should be between 3 and 8 pages excluding references. Different projects may take different numbers of pages to describe, and it depends on whether you're working by yourself or in a group. If you have lots of analysis and discussion or are trying something more ambitious, your paper might be longer; if you're implementing something complex but succinctly described, your paper might be shorter.

Your project is *not* graded solely on the basis of results. You should approach the work in such a way that success isn't all-or-nothing. You should be able to show results, describe some successes, and analyze

[2]Style files available here: `http://www.acl2019.org/EN/call-for-papers.xhtml`

[3]The Iyyer et al. paper is a good example of this: `https://people.cs.umass.edu/~miyyer/pubs/2015_acl_dan.pdf`

why things worked or didn't work beyond "my code errored out." Think about structuring your work in a few phases so that even if everything you set out to do isn't successful, you've at least gotten something working, run some experiments, and gotten some kind of results to report.

Grading: We will grade the projects according to the following rubric:

- **Scope (25 points)**: Is the idea of sufficient depth for a course project? While your idea or fix does not have to work wonderfully, you will lose points here if all you can show is shallow analysis of the base system.
- **Implementation (25 points)**: Is the implementation described reasonable? Is the idea itself technically sound? You might lose points here if we perceive there to be a technical error in your approach. For example, perhaps you tried to modify neural network training in a way that is totally unconnected to your stated goal, or your modification was otherwise erroneous.
- **Results/Analysis (25 points)** 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? What aspects of the data/model might not be right? There are a few things you should report here: **Key results**: You should report results from a baseline approach (your initial trained model) as well as your "best" method. If doing your own project, baselines such as majority class, random, or a linear classifier are important to see. **Ablations**: If you tried several things, analyze the contribution from each one. 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.
- **Clarity/Writing (15 points)**: Your paper should clearly convey a core idea/hypothesis, describe how you tested it/what you built, and situate it with respect to related work as best you can. **Abstract and Introduction**: Did you provide a clear summary of the motivation, methodology, and results? **Method**: Is the presentation of what was done clear? **Results**: Is the work experimentally evaluated? Are there clear graphs and tables to communicate the results? Don't just inline 1-2 numbers; make your analysis more detailed than that.

Extra Credit: You can earn 2 points of extra credit by filling out the eCIS course evaluation. **Take a screenshot of the page showing that you've completed the survey (not your response itself, which is confidential) and upload it along with your final project.**

4 Compute and Feasibility

Large neural net methods can be slow to train! Training a model on even 10,000 QA examples can take hours (when running only on CPU). Keep this in mind when deciding what kind of project to do.

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 5-15 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.

Using Checkpoints The HuggingFace trainer checkpoints the models periodically. As a result, you can start a long training run, leave it going for several hours, 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 November 2021, it's $9.99 per month.

References

[Bartolo et al.2020] 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.

[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.

[Chen and Durrett2019] 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, June. 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)*.

[Dua et al.2021] Dheeru Dua, Pradeep Dasigi, Sameer Singh, and Matt Gardner. 2021. Learning with instance bundles for reading comprehension.

[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*.

[Glockner et al.2018] Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for*

Computational Linguistics (Volume 2: Short Papers), pages 650–655, Melbourne, Australia, July. 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.

[Kaushik and Lipton2018] Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium, October-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.

[McCoy et al.2019] Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, July. 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.

[Rajpurkar et al.2016] 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.

[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.

[Wallace et al.2019] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. 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 2153–2162, Hong Kong, China, November. Association for Computational Linguistics.

[Williams et al.2018] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June. 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.

[Yang et al.2018] 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, October-November. 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.