CS378 Natural Language Processing: Final Project

Proposal Due Date (for independent projects ONLY): Tuesday, February 21rd at 11:59pm CST
Check-In Due Date: Tuesday, April 4th at 11:59pm CST (2.5 week before final deadline)
Final Report Due Date: Thursday, April 27th at 6:00pm CST
NO slip days allowed for final project, all submissions on Gradescope

Collaboration You can work on the final project in teams of two (encouraged) or individually. For a group project, all partners will receive the same grade for it, so work should be distributed evenly. Individual projects can be less ambitious in terms of scope but should not be less complete.

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 (including human resources) used must be clearly cited. Unlike for homework assignments, you are free to use AI assistance if you like. If you do, however, clearly document your usage in your final report with a separate heading titled "AI Assitance".

Compute Resources The course has an allocation of GPU resources on the maverick2 cluster on TACC. You can use these resources to run any intensive training jobs for your project. Keep in mind that these resources are limited and choose a final project that doesn't have an astronomically high compute budget.

Running jobs on clusters is slightly different from running on your local machine. You typically login via a login node, setup your code and dependencies (Anaconda, Python, Pytorch) on it and then issue a command to submit a job to the cluster. The cluster then schedules this job once a compute node (with GPUs) is available. The nodes share a networked filesystem and the results (outputs, errors and other files) are written to it once the job completes. These can be viewed from the login node.

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 early 2022, it costs $9.99 per month.

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 language models (Devlin et al., 2019) offer competitive 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. NLI: The Stanford NLI dataset (Bowman et al., 2015), MultiNLI (Williams et al., 2018)
2. QA: SQuAD (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), MRQA mixed QA dataset (Fisch et al., 2019)

You are welcome to try other tasks or datasets if you prefer. **You will analyze the model's performance and shortcomings, try to improve it, and describe whether what you did fixed things.** If you are working individually for this option, you are still expected to complete the project as described below. Compared to two person option, however, your analysis can be less extensive and your fix can be simpler.

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 1.1 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.

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).
- Removing incorrectly labeled data from the training dataset (Chen 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, but you should make this clear in your final report. 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.** Please look at this reproduction challenge if you are interested in this option.

When you are designing a fix, you should think about your evaluation plan. How can you empirically show that your fix brings desired effects? **It will be very hard to get overall much stronger performance on an in-domain test set (and you are not required to achieve this)**, but we expect that well-implemented projects should be able to show some improvement either on a subset of examples that the fix was targeting or generalization to different distribution. 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!

1.1 Getting Started

Installation instructions Please follow the instructions in the GitHub repository here:
`https://github.com/utcsnlp/cs378_fp`

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`. You do not have to use all of the starter code.

HuggingFace The skeleton code is heavily based on HuggingFace `transformers` library, 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.

1.2 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.

1.3 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 analyzing why it fails. Just saying "I tried X and wrote 100 lines of code but it still crashes" is not a good project outcome – your code might not achieve desired results, but it should be bug free. 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 modification.

1.4 Delieverable

Check in (due April 4th, 10% of grade) You should turn in a check-in (at least a couple paragraphs, no more than 1 page) on check-in due date. 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"). Your check-in document should sketch a plan for the your project, but will mainly be graded that. You should assess your current progress and required future work.

Final Report (due Thursday, April 27th at 6:00pm CST, 90% of grade) See section 3.

2 Independent Research Project

For this option, you (and your teammate) will decide on project topic of your choice! You can choose any topic in NLP (either covered or not covered in the class), but 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. 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 (please ask course staff if you are unsure). Refer to ACL proceedings for inspiration. You can focus on either:

- A new model architecture for existing problems (or a variant/extension of an existing model)
- A new training, optimization, or evaluation method for existing problems
- A new application of NLP technology – here, you will apply an existing model to a new task. Please motivate the task carefully.
- Experimental and/or theoretical analysis of datasets, approaches, or models.

You can find a list of existing datasets from the following lists:https://nlpprogress.com/, https://www.kaggle.com/datasets, https://huggingface.co/docs/datasets/

Scope of the project It is crucial to scope the project well. The expected scope should be similar to the first option (dataset artifact project). Your goal is not necessarily to come up with the best model that can beat the state-of-the-art performance. Whether your model is good or not, you should evaluate your model and aim to provide an explanation for the behaviors of your model. You can look at final projects from similar courses for inspiration here. Be mindful of the computational resources required for your project.

2.1 Deliverables

Project Proposal (due February 21rd, 10% of grade) You will submit a two page project proposal on the due date. You can use whatever format for the project proposal and checkin document, but the page estimate is based on the final report format provided below. The proposal should include the followings:

- Title
- Team members (and their EIDs)
- Paper summary (roughly 1 page): Your project will likely build on existing work. You should select a paper that seems most relevant to your final project. For example, if you are doing a reproduction study, you will describe that paper. For an independent study, if you are introducing a variant of a model, you should select a paper that either describes the model that you are trying to improve. If you are introducing a new task, you can select a paper that describes a similar task but different from your own. There is no restriction on what paper this should be, but it would make your job easier if you choose a high quality paper (peer-reviewed papers (already accepted at conference) is good, but some recent papers (yet to be published) can be also good). In this paper summary, you should include **why you have chosen this paper**, **contributions of this paper** and **limitations of the paper**. You should summarize the paper succinctly, such that your classmates can understand the gist of the paper. It will

be helpful to provide background of the paper in the context of research community (how this relates to other work, why this is an important paper, etc).

- Project description (roughly 1 page): Here you will describe a rough plan for your project. You are not bound to this document, and things can progress differently from your initial plan (at the end, this is a mini research project!). This part should describe **the main goal of the project**, **evaluation plan (including training / evaluation data and evaluation metric)**, and **baseline models**.

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.

Check in (due April 4th, 5% of grade) This should be about one page, describing what you have achieved so far (and setbacks you have encountered), and plan for the final report. 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., "We have run baseline models and now moving on to making modifications to the model, etc").

Final Report (due Thursday, April 27th at 6:00pm CST 85% of grade) See below!

3 Final Report Instruction

Here is a general guideline about final project report, that will be applicable for both options. You will not submit code but include a link to a public github repository of your code in the final report. You can find the skeleton document for final report with grading rubrics for option 1 and option 2. You can copy the skeleton project, and complete your own by replacing to-dos with your contents. We highly recommend reading the final report skeleton document **early** to get a sense of what will be expected.

Here we list some sample papers to get a sense of how great final report should look like: Deep Averaging Network, EndtoEndCoref. Just like conference papers, it should begin with an abstract and introduction, clearly describe the proposed idea or proposed reproduction, present technical details, give results, compare to baselines, provide analysis and discussion of the results, and cite sources throughout (you'll probably want to cite at least 10 papers depending on how broad your topic is). Here are a few aspects that will be considered.

Scope of the project: Is the idea of sufficient depth for the final project? While your idea does not have to work wonderfully, you will lose points here if all you can show is shallow analysis of the base system. **Technical Soundness**: Is your analysis, solution and evaluation technically sound? Your experimental set up should be fair for your model and the baseline model being compared to, and modifications should match your described motivation. **Clarity**: 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.

The maximum page limit is 8 pages excluding references. Longer report is not necessarily better, and the length will depend more on the nature of your project. For example, 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.

Credit: The dataset artifact project was developed by Greg Durrett.

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.

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.

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.

Derek Chen, Zhou Yu, and Samuel R. Bowman. 2021. Learning with noisy labels by targeted relabeling. *ArXiv*, abs/2110.08355.

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.

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)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

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

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. *ArXiv*, abs/1910.09753.

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.

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*.

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 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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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*.

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.

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.

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.

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.

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.

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.

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.