CS388 Final Project

Proposal Due: March 11, 11:59pm
Presentations: May 4 and May 6, in class
Final Report Due: May 12, 11:59pm

Collaboration You are free to work on this project in teams of two (strongly 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 may collaborate with a person from outside the course as well in case you're also using this final project for another course. 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.

Combining with other final projects You are allowed to combine this project with your research or projects from other courses. However, your project must still involve concepts from this course! You are allowed to apply these models to data that isn't language data provided that it has some interesting language-like structure (e.g., genomics data, time-series data, etc.). Investigating feedforward neural network architectures on MNIST would *not* be an acceptable course project.

Overview

This project is an independently-conducted study of NLP. You have two options. The first is to pursue original research on an NLP problem, and the second is to attempt to reproduce results from a prior paper. The final project is worth 40% of your course grade.

Original Research There are a several possible approaches here. You might do a more engineering-style project: pick a task and a dataset, design or expand on some model, and try to get good results, similar to what you were doing in the other projects in the course. You can also do a more analytical project: pick some problem and try to characterize it in greater depth. What does the data tell us? What does this tell us about language or about how we should design our NLP systems? What can interpretation techniques, contrast sets, or other focused evaluation measures tell us about how models are doing?

Your end goal for this option shouldn't be just reimplementing what others have done. However, implementing someone else's model or downloading and running an existing model are great first steps and might end up getting you most of the way there, and implementing a couple of approaches in order to gain some insight from comparing them can be a good project.

This project is *not* graded on how well your system works, as long as you can convincingly show that your model is doing something. Start with baby steps rather than implementing the full model from scratch: build baselines and improve them in a direction that will eventually take you towards your full model. You should think these steps through in your proposal.

Reproduction The goals here follow the ML Reproducibility Challenge.[1] You should pick a prior paper (it can be from any year, although more recent and less-tested methods are more likely to yield surprising results) and evaluate how well you can reproduce the results of the paper. This involves several steps:

[1] `https://paperswithcode.com/rc2020/task`

1. Figure out what results you want to reproduce.

2. Figure out what code is available, and see if you can get it running easily. If you can get it running in 30 minutes, that will change the scope of your project compared to code that might take hours to resolve or intervention from the authors.

3. Decide what you want to focus on in your reproduction. To quote from the challenge: *Just re-running code is not a reproducibility study, and you need to approach any code with critical thinking and verify it does what is described in the paper and that these are sufficient to support the conclusions of the papers. Consider designing and running unit tests on the code to verify it works well and as described. Alternately, the methods presented can also be fully re-implemented according to the description in the paper.*

We will hold reproducibility papers to a high standard! In particular, you should do exploration and experimentation on par with what's expected from the original research option. Don't pick a paper where you can reproduce the network in 20 lines of PyTorch, tune hyperparameters, put your results in a table, and declare victory.

Deliverables

Proposal (5 points) You should turn in a **one page proposal** on the proposal due date. This proposal should outline what problem you want to address or what paper you're reproducing, what dataset(s) you plan to use, and a rough plan for how you will pursue the project. While you don't need a full related work section, you should mention any relevant prior work and state how your project relates to it. The course staff will then provide feedback and guidance on the direction to maximize the project's change of succeeding.
Grading: 5 points for turning in a proposal meeting a minimum level of coherence and quality. You are not evaluated on how good the idea is—this is a stage to get feedback and refine things.

Final Report (90 points) The primary deliverable is a paper written in the style of an ACL/NeurIPS/etc. conference submission. 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 5-10 papers depending on how broad your topic is).

If you are working in a team of two, the paper should be on the order of 8 pages excluding references; working alone, you should target more like 4-6 pages. Don't treat these as hard page requirements or limits, and let the project drive things. 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.

Critically, 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 why things worked or didn't work beyond "my code errored out." Think about structuring your proposal in a few phases (like the projects) 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:

- **Clarity/Writing (20 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.

- **Implementation/Soundness (35 points)**: Is the idea technically sound? Do you describe what seems like a convincing implementation? Is the experimental design correct?
- **Results/Analysis (35 points)** Whether the results are positive or negative, try to motivate them by providing examples and analysis. If things worked, what error classes are reduced? If things didn't work, why might that be? What aspects of the data/model might not be right? If you're writing a paper that revolves around building a system, you should try to report results for a baseline from the literature, your own baseline, your best model, and possibly results of ablation experiments.

Final Presentation (5 points) During the last week of class, every group will give a 3-to-5-minute presentation on their project (depending on the number of groups). This presentation should state the problem, describe the methodology used, and give highlights of the results. Because the project reports won't have been due yet, these results might be preliminary, but should be nonzero. Teams will be assigned a presentation date randomly at the time the proposal is due.
Grading: The final presentation should be clear and fit within the allotted time limit. It should describe your methodology, related prior work, and preliminary results or analysis. Most students will receive 5/5 on this unless the presentation is somehow incomplete.

Choosing a Topic

The following is a (non-exhaustive!) list of possible directions for either new research or reproductions, just a few to give you some pointers. Another approach is to look through the papers in recent ACL/EMNLP conferences[2] and see if there are topics that seem interesting to you, then try to find datasets for those tasks.

Interpretability/Probing Neural Networks Given the success of neural models, particularly BERT, there is increased interest in understanding them: what their representations capture, how they generalize, etc. For example, we can using probing tasks to analyze the abilities of LSTMs to generalize along certain dimensions (Linzen et al., 2016) or to understand what the layers of BERT capture (Tenney et al., 2019). One viable project option is to try to improve our understanding of these models through new analyses or probing them in new ways. Note that with such projects, you should really be aiming to test a clear hypothesis and be able to accept/reject it based on your results. It's not a good project to just say you'll plot some aspect of BERT, then plot it and make handwavy conclusions about things. For more inspiration, see the proceedings of the recent Blackbox NLP workshops at ACL 2019 and EMNLP 2020.

Contrast Sets Some recent work has proposed to explore models' behaviors using carefully crafted sets of related examples, called contrast sets (Gardner et al., 2020) or counterfactually-augmented data (Kaushik et al., 2020). These approaches take "base" data points and tweak them in targeted ways that may change or not change the label, such as changing some of the positive adjectives in a sentiment example to be different adjectives. We can then see whether the model does the right thing on this new example, allowing us to understand whether the model is "right for the right reasons" or relying on spurious clues. A related idea is "checklists" to test the behavior of models (Ribeiro et al., 2020). All of these approaches provide ways to take a problem and study some phenomenon about that problem more deeply: are current approaches able to model the problem correctly or not?

[2] https://www.aclweb.org/anthology/

Zero-shot capabilities of pre-trained LMs While we have focused extensively on what pre-trained models can do after fine-tuning, there is also active investigation into what these models can do organically, with no training. One benefit of this is that you can do it without re-training these models. Just running inference is fairly tractable with limited resources. Past work has looked at testing them to understand what explicit knowledge they contain in English (Petroni et al., 2019) or in other languages (Kassner et al., 2021). Other recent work has explored few-shot learning with a combination of patterns that help the model better adapt to the new task in a GPT-3-style fashion (Schick and Schütze, 2020) (this does involve fine-tuning, but it's small scale). We caution that many promises of incredible behavior only show up in the very large-scale language models and are probably not as impressive as they sound, but nonetheless, think about what you might do along these lines.

Other domains and languages Tasks like POS tagging, NER, sentiment analysis, and parsing are well understood and have been thoroughly studied; it is hard to improve on state-of-the-art models for these on English datasets. However, other domains (web forums, biomedical text, Twitter), and other languages are less well understood, but datasets exist for these and there are small "cottage industries" of papers around each of these topics. Many of the state-of-the-art English systems for these tasks have been discussed in class—perhaps download these and see how they compare to other models on new data.

Language and Code Classic semantic parsing on datasets like Geoquery is a bit hard to advance due to small datasets and limited domains. However, there are plenty of interesting language-to-code style tasks that are in a similar domain. There exist a plethora of datasets for the text-to-SQL task (Finegan-Dollak et al., 2018), which you can investigate. The CodeXGLUE[3] benchmark is a recent resource aggregating many language-and-code tasks you may wish to consult.

Multilingual models Pre-trained models like ELMo and BERT have been extensively studied for English. Because of the word piece abstraction, there is clear transfer to related languages that use Latin script. Because of code-mixed data, these models also have some success for frequent languages that don't share an alphabet with English, such as Chinese. However, for more distant languages like Thai which have their own script, these models underperform. Past work (Pires et al., 2019) has some analysis of this on a few basic tasks, but there's a lot more to investigate here.

Computational Linguistics While we haven't focused on it much in this class, if you want to use any of the models in this course to study phenomena in language, you are more than welcome to!

CAUTION Your project probably should **not** revolve around needing to fine-tune large language models on large-scale datasets (100k+ examples). Even getting a very basic version of this working is tough without access to significant other compute resources. Needing to train a model like this once is okay, but expecting to do it 10+ times and iterate on your results will prove challenging unless you have a lot of experience with these types of systems. Machine translation and summarization rely on training on particularly large datasets. There are good projects you can do in these domains, but you may wish to focus on low-resource settings or more traditional models, as large-scale neural approaches won't be feasible to explore unless you have access to significant GPU resources.

Fine-tuning BERT-Base on a modest-sized dataset (less than 10k examples) can often be done effectively with more limited resources, but will still typically require GPUs.

[3] https://github.com/microsoft/CodeXGLUE

Computational Resources Available

This course has an allocation on the Maverick2 cluster on TACC. Contact us if you wish to know more abut this.

Google Cloud Platform is also a way to get GPU time, either for free when you sign up (although the nature of this promotion changes sometimes) or for relatively inexpensive (a couple of large experiments may cost $20).

Submission

You should submit your final report in a single PDF on Canvas. No other datasets, code, results, etc. need to be uploaded.

Slip Days Slip days may not be used for any component of this project.

References

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh 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. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online, November. Association for Computational Linguistics.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual LAMA: Investigating Knowledge in Multilingual Pretrained Language Models. *arXiv cs.CL 2102.00894*.

Divyansh Kaushik, E. Hovy, and Zachary Chase Lipton. 2020. Learning the Difference that Makes a Difference with Counterfactually-Augmented Data. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. In *Transactions of the Association for Computational Linguistics*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How Multilingual is Multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

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.

Timo Schick and Hinrich Schütze. 2020. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. *arXiv cs.CL 2009.07118*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.