CS388 Final Project

Proposal Due: February 20, 11:59pm
Check-ins: April 4, 11:59pm
Presentations: April 23 and April 25, TBD
Final Report Due: May 3, 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 some problem in 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.

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? Most contributions appropriate for an NLP/ML conference or workshop paper are in scope here.

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. Implementing a couple of approaches in order to gain some insight from comparing them can be a good project (somewhere between research and reproduction, and that's fine!). **For projects in this area, you should start with literature search and include that in your project proposal to make sure you're not missing relevant prior systems.**

This project is *not* graded on how good your results are, as long as you can convincingly show that you've done something: trained an effective model, collected some quality data, etc. Start with baby steps rather than implementing your full approach from scratch: build baselines and improve them in a direction that will eventually take you towards your full approach. You should think these steps through in your proposal.

Reproduction The goals here follow those of 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. 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. Try to have an interesting question that you can answer in a deep way.

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, prior work, and a rough plan for how you will pursue the project. While you don't need a full related work section, you should mention the most relevant prior work you've found 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.

Check-in (15 points) You should turn in a check-in on the check-in due date. This should probably be 2-4 pages in length and look like a fleshed out version of the proposal, building towards your final project. You should respond to feedback from the proposal stage, include a more detailed discussion of related work, and most critically, *have some kind of preliminary results for your approach.* Think about what you can do to make sure you have something to show at this stage, even if it's more unit test-style validation that your code is correct.
Grading: You will lose points on the check-in if we perceive you have not dedicated time to flesh out the project beyond the proposal and take a first stab at the work on it.

Final Report (70 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

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

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 might 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. If you have lots of analysis and discussion yielding full-page figures 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 grade projects starting from a "base score" of 60/70 for projects that are executed according to the specification laid out here. This score will then be moved up or down based on the following criteria:

- **Scope**: Was the scope appropriate for the project? You will gain or lose points depending on our subjective assessment of the overall work done and whether it fits the charge you were given.
- **Clarity/Writing**: 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. Doing a thorough job of this will raise your grade. Your grade will be lowered if you fail to explain what you did in a clear fashion or do not present results in a coherent way.
- **Implementation/Soundness**: Is the idea technically sound? Do you describe what seems like a convincing implementation? Is the experimental design correct? You may lose points here if
- **Results/Analysis** 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. If you're doing a reproduction, try to be as thorough as you think is appropriate.

Final Presentation (10 points) During the last week of class, every group will give a 3-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.

You will want to keep this presentation focused on high-level aspects of your approach, methodology, and results/examples of the output.

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.

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/NeurIPS/etc.

conferences[2] and see if there are topics that seem interesting to you, then try to find datasets for those tasks.

Controlled Generation There is a lot of excitement around "prompt engineering", or how you can elicit behaviors from large language models like GPT-3 with the right prompts. However, this is a very blunt instrument compared to the rich range of inference strategies available. We have discussed approaches for inference like beam search and nucleus sampling. There are other more sophisticated approaches like "neurologic" decoding (Lu et al., 2021), FUDGE (Yang and Klein, 2021), diffusion models (Li et al., 2022b),[3] contrastive decoding (Li et al., 2022a), and more. All of these papers exhibit some interesting and diverse generation tasks. Many of these approaches can be implemented on smaller LLMs like Llama.

Probing Neural Networks, Mechanistic Interpretability 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 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.

A recent flavor of this is "mechanistic interpretability." Neel Nanda's post with a list of open problems is also a good starting point.[4] For these projects, you should likely aim to devise experiments that will prove something one way or another, rather than go "fishing" for something like a magic attention head that explains a certain behavior, which you may not be able to find.

Understanding ICL In-context learning has been extensively studied in recent years but is still not fully understood. If you pursue this, you may be able to get somewhere with smaller Llama-scale models, but you will probably want to explore models like those on OpenAI, since the biggest models are usually significantly better at this. You can follow on the papers discussed in the "Understanding ICL" lecture.

Chain-of-thought Similarly, you can follow on the papers discussed in the "Chain-of-thought" lecture. A lot has been explored here

Inference in LLMs A large number of recent approaches such as speculative decoding (Leviathan et al., 2023) and Medusa[5] have been proposed for speeding up LLM inference. Inference efficiency can be a nice place to work as it only requires being able to run LLMs rather than do expensive pre-training fine-tuning, and can often be done on a local machine (and improving the throughput locally can even be your goal!). You can explore and extend existing methods that are out there or try to devise your own! One way to approach this might be to think of a workflow or application (e.g., generating json-formatted data) and think about whether there are optimizations specific to that application you can explore.

[2]`https://www.aclweb.org/anthology/` for papers at the NLP ones; you can also look at individual conference websites which will have programs organizing papers by topic area.

[3]Note that despite the very strong results for image synthesis, the results in this paper are not that strong and are far behind pre-trained methods (this is not a pre-trained model). Don't just propose something related to diffusion models because you think they'll work great—you will likely struggle a lot to get them working.

[4]`https://www.lesswrong.com/posts/LbrPTJ4fmABEdEnLf/200-concrete-open-problems-in-mechanistic-inter`

[5]`https://www.together.ai/blog/medusa`

Knowledge in Language Models and Continual Learning There has been significant work looking at the ability of language models to recall factual knowledge in English (Petroni et al., 2019) or in other languages (Kassner et al., 2021). However, this knowledge may become outdated (Onoe et al., 2022) and it is unclear how to address this. Naive approaches include continuing to train models on new data (Gururangan et al., 2020), but this continual learning approach is still being benchmarked and evaluated (Jang et al., 2022). At the same time, other approaches have been proposed for editing knowledge in models (Meng et al., 2022; Mitchell et al., 2022). Projects in this space could focus on benchmarking current models' capabilities in certain settings, understanding how they learn or how knowledge can be updated over epochs of continual learning, what sorts of knowledge can be added or edited most effectively, and more.

Parameter-efficient Fine-tuning With the rise of large models, there is a lot of interest in methods for fine-tuning these models without updating all of their parameters. There are methods such as LoRA (Hu et al., 2021), prefix tuning (Li and Liang, 2021), and plenty of others; see a survey by Houlsby et al. (2019). **These can either be used as a component of other projects to make things more efficient, or you can study them themselves.** Developing new methods may be hard, but you can see if they enable you to tackle other problems in some creative ways!

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. Perhaps try getting access to the latest systems in these areas and see how they perform.

Language and Code Models like Copilot, Codex, and CODE-DAVINCI-002 have strong abilities when it comes to source code completion. There is lots of work at the intersection of language and code: either treating code as language and using language models to complete it, looking explicitly at prompts/comments, solving coding challenge problems (like Google's Minerva), and more. Ideas for projects here could be about new applications for them, ways to improve how they function, or doing smaller-scale tasks with smaller-scale models like CodeT5 (Wei et al., 2023).

Applications There are many interesting applications of methods from this course to different problems: hate speech detection, review summarization, open-domain question answering, and beyond. This is a broad category and projects here may feature many different challenges. In all cases, you'll want to decide what's challenging about your project; is it some aspect of data collection? Evaluation? **Just running a model and reporting results on a dataset isn't sufficient for a good project.** Also note that we either want to see new work *or* a reproduction study, so for whatever application you are interested in, you should look carefully into the literature to see what existing approaches are applicable.

Multilingual models There is a massive growth in multilingual models like XLM-R, mT5, BLOOM, and more. Partially because of things like code-mixed data, these models also have some impressive successes at learning cross-lingual representations, even for 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), which might be the case for projects involving SFT/RLHF. 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 and access to good GPU resources.

Computational Resources Available

This course has an allocation on the Lonestar6 cluster on TACC. Contact us if you wish to be added to it.

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).

Colab Pro (for $9.99/mo) is also an option that makes GPUs available to you.

Unfortunately, we are not able to provide access to LLM APIs like OpenAI or Cohere. However, many projects may be doable for only modest cost (less than $20 in expenditures). You can also use ChatGPT, though note that it may be unreliable and the lack of an official API makes it hard to run on standard test sets. Do not wait until late in the project to sort access issues out, as availability of all of these services is subject to change.

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

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *arXiv*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *arXiv*.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. 2022. Towards Continual Knowledge Learning of Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast Inference from Transformers via Speculative Decoding. *arXiv*, abs/2211.17192.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August. Association for Computational Linguistics.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022a. Contrastive Decoding: Open-ended Text Generation as Optimization. In *arXiv*.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022b. Diffusion-LM Improves Controllable Text Generation. In *arXiv*.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2021. NeuroLogic A*esque Decoding: Constrained Text Generation with Lookahead Heuristics. In *Proceedings of NAACL*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Associations in GPT. In *Proceedings of NeurIPS*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. Fast Model Editing at Scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yasumasa Onoe, Michael J.Q. Zhang, Eunsol Choi, and Greg Durrett. 2022. Entity Cloze By Date: What LMs Know About Unseen Entities. In *Findings of NAACL (short)*.

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)*.

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)*.

Jiayi Wei, Greg Durrett, and Isil Dillig. 2023. TypeT5: Seq2seq Type Inference using Static Analysis. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled Text Generation With Future Discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.