Final Project

CSE 5525

Important Dates

- **Proposal Due**: October 3, 11:59pm
- **Check-ins**: TBD
- **Presentations**: TBD
- **Final Report Due**: TBD

1 Collaboration

You must work in teams of 2 or more. Smaller teams can be less ambitious in scope but should not be less complete: a half-implemented system does not make a good project outcome. All partners are expected to contribute equally to the submission. You are required to detail all your contributions in your reports which will determine your grade. 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.

2 Combining with Other Projects

You may combine this project with other research or coursework projects, provided that it is aligned with NLP concepts. Non-language data is allowed if it has language-like structures (e.g., genomics, time-series data). Simple feedforward neural networks on MNIST are not acceptable.

3 Overview

This project is an independently conducted original study of some problem in NLP. 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 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

4 Deliverables

Proposal (10 points) You should turn in a 1-2 page proposal (using a Latex Template[1]) on the proposal due date. This proposal should outline what problem you want to address, 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. We will then provide feedback and guidance on the direction to maximize the project's change of succeeding.

Grading: 10 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 that 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 (65 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 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).

The paper should be the order of 8 pages excluding references; 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 55/65 for projects that are executed according to the specification laid out here. This score will then be moved up or down based on

[1] LaTex template: https://github.com/acl-org/acl-style-files

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?
- 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.

4.1 Final Presentation (10 points)

During the last week of class, every group will give a 3-5 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. 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.

4.2 Choosing a Topic

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

- **Controlled Text Generation**
  Text generation models often produce generic or uncontrolled outputs. This project focuses on techniques to control text generation based on attributes such as sentiment, formality, style, or topic relevance. Possible approaches include reinforcement learning, contrastive decoding, and attribute-based prompt engineering. You can also explore external control signals, such as conditioning models on structured data or linguistic constraints. The project can evaluate control effectiveness using automated metrics and human evaluations.

- **Neural Network Interpretability in NLP**
  Despite their impressive performance, large language models (e.g., BERT, GPT) remain difficult to interpret. This project aims to analyze how these models represent linguistic structures and whether they encode human-interpretable concepts. Techniques such as attention visualization, probing classifiers, and layer-wise relevance propagation can be used to study what different layers of a model learn. The goal is to improve our understanding of NLP models, detect biases, and enhance their trustworthiness in high-stakes applications.

- **Evaluating and Improving In-Context Learning**
  Large language models exhibit emergent in-context learning abilities, allowing them to perform new tasks without parameter updates. This project explores different prompt engineering strategies, such as chain-of-thought reasoning, few-shot learning, and retrieval-augmented prompting. You can evaluate how different model sizes and architectures affect in-context learning performance across a variety of NLP tasks, including question answering, commonsense reasoning, and classification. The project may also investigate failure cases and strategies for improving prompt effectiveness.

- **Parameter-Efficient Fine-Tuning (PEFT)**
  Fine-tuning large language models is computationally expensive, but parameter-efficient fine-tuning (PEFT) methods, such as LoRA, adapters, and prefix tuning, enable efficient adaptation. This project investigates the trade-offs between full fine-tuning and PEFT approaches in terms of performance, parameter efficiency, and generalization. You can conduct experiments on a range of NLP tasks to determine when PEFT methods work best and analyze whether they retain knowledge from the base model while improving specialization.

- **Knowledge Editing in Language Models**
  Language models are known to encode factual knowledge, but this knowledge can become outdated or inaccurate. This project focuses on methods to update or remove specific facts from models without full retraining. You can experiment with retrieval-augmented generation, memory-augmented networks, or direct parameter modifications to assess their effectiveness in knowledge editing. The evaluation can involve testing whether updates propagate consistently across different model behaviors while preserving other knowledge.

- **Multimodal NLP (Text + Audio/Image)**
  Multimodal learning enables models to process multiple data types, such as combining text with images or speech. This project explores how NLP techniques can be applied to multimodal problems, such as image captioning, visual question answering, or speech-to-text translation. You can fine-tune existing models like CLIP, multimodal transformers, or develop custom architectures that leverage both textual and non-textual information. Evaluation methods may include accuracy metrics, qualitative assessments, and human-in-the-loop evaluations.

- **NLP for Low-Resource Languages**
  Many state-of-the-art NLP models are trained primarily on high-resource languages, leaving low-resource languages underrepresented. This project investigates transfer learning, multilingual embeddings, and zero-shot/few-shot adaptation techniques for improving NLP performance on underrepresented languages. Possible directions include evaluating pre-trained multilingual models (e.g., mBERT, XLM-R), developing tokenization strategies, or curating domain-specific datasets for benchmarking. The goal is to bridge the language gap and improve accessibility.

- **NLP for Social Good**
  NLP has the potential to address real-world problems such as misinformation detection, mental health analysis, and disaster response. This project explores socially beneficial applications of NLP, such as detecting toxic language, summarizing crisis-related news, or classifying sentiment in mental health forums. Ethical considerations, such as bias mitigation and fairness, will also be an important aspect of the project. You can develop models using real-world datasets and assess their performance in socially relevant contexts.

- **Task-Specific Prompt Optimization**
  Prompt engineering is crucial for improving the effectiveness of large language models on various NLP tasks. This project investigates different prompt construction techniques, such as zero-shot learning, few-shot learning, and self-refinement strategies. You can experiment with adversarial prompts to analyze model robustness or use meta-learning to automate prompt selection. A key outcome of this project is understanding how different prompt structures influence model behavior and performance.

- **Efficient NLP Model Inference**
  Deploying NLP models in real-time applications requires optimizing inference speed and memory usage. This project explores techniques such as quantization, knowledge distillation, speculative decoding, and retrieval-based optimizations to reduce computational overhead. The goal is to benchmark various efficiency-improving techniques and analyze their trade-offs between speed, accuracy, and interpretability. Applications may include chatbots, automated summarization, and real-time speech-to-text systems.

- **Continual Learning for NLP Models**
  Unlike static training, continual learning enables NLP models to adapt to new data without catastrophic forgetting of prior knowledge. This project explores lifelong learning strategies such as replay-based memory, dynamic architectures, and modular adaptation. You can experiment with evolving datasets, such as news streams, and analyze how models retain previously learned knowledge while incorporating new information.

- **Adversarial Attacks and Defenses in NLP**
  NLP models are vulnerable to adversarial attacks, such as small perturbations in text that cause misclassification. This project investigates attack strategies, including character-level, word-level, and paraphrased adversarial examples, as well as defenses such as adversarial training and input denoising. You can evaluate how robust different architectures (e.g., transformers, LSTMs) are against adversarial manipulations and propose improvements to enhance model resilience.

- **Bias Detection and Mitigation in NLP Models**
  Many NLP models exhibit biases related to gender, race, and political viewpoints due to skewed training data. This project focuses on detecting and mitigating biases in NLP models using adversarial training, debiasing word embeddings, or data augmentation techniques. A thorough analysis can be conducted on datasets with known biases, and experiments can test interventions to promote fairness in language model outputs.

- **Code Understanding and Generation**
  With the rise of AI-driven coding assistants like Codex and CodeT5, NLP techniques are increasingly used for code completion, generation, and documentation. This project explores how well these models understand programming languages, whether they generalize across

tasks, and how they compare to human-written code. You can benchmark models on tasks like code summarization, bug detection, and automated documentation generation.

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.

5 Computational Resources

This course has an allocation on OSC. Please find information on Teams on how to add yourself. Contact us if you face any issues. Google Cloud Platform or Microsoft Azure 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.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 Anthropic (they might be available through Azure AI with free credits). 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.

6 Submission Guidelines

You should submit a single PDF for proposals on Gradescope.

For other submissions, details TBD.

7 Late Days

Late days cannot be used for any project component.