CS388 Final Project

Proposal Due: October 15, 11:59pm
Presentations: December 3/5, 12:30pm
Final Report Due: December 13, 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.

Deliverables

This project is an independently-conducted study that constitutes original research on an NLP problem. The final project is worth 40% of your course grade. The deliverables are as follows.

Proposal (10 points) You should turn in a **one page proposal** on the proposal due date. This proposal should outline what problem you want to address, what dataset(s) you plan to use, and a rough plan for how you will pursue the project (e.g., "we propose to download X system, run it, then implement our system on top of their framework and compare the results"). While you don't need a full related work section, you should mention a few pieces of prior work and state how your project relates to them. The course staff 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.

Final Report (80 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, 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 5-6 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.

Note that your project is *not* graded solely on the basis of results. You should feel free to try an idea that's a bit "out there" or challenging as long as it's well-motivated and can be evaluated. Critically, you should also 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 (16 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. See the "Tips for Academic Writing" on the course website if you have doubts about what is expected.
- **Implementation/Soundness (32 points)**: Is the idea technically sound? Do you describe what seems like a convincing implementation? Is the experimental design correct?
- **Results/Analysis (32 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 (10 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.

Choosing a Topic

There are a few directions you can go with this project. 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 first three projects. 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?

Your project should include *something novel*: 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, and implementing a couple of approaches in order to gain some insight from comparing them can be a good project. One good way to attack things is to pick a task and a dataset, download and run a model from the literature, and assess the errors to see what it does wrong. While it's best to go in with some intuition of how you can improve things, letting yourself be guided by the data and not sticking to assumptions that may prove incorrect is the best way to build something that actually works well.

Be bold in your choice! 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. The initial projects in this class are structured to do this, to give you an example of this process.

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

Text annotation tasks 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.

Entity Linking Entity linking involves resolving a span of text in a document (*John Smith*) to a Wikipedia article capturing that entity's true identity (`https://en.wikipedia.org/wiki/John_Smith_(explorer)`). Classical methods use data from Wikipedia and use features such as cosine similarity of tf-idf vectors between the source context and target Wikipedia article (Ratinov et al., 2011). A newly released dataset (Eshel et al., 2017) is much cleaner and larger and more admissible to training neural network models. Multilingual approaches (Sil et al., 2018) might also be nice to investigate or follow up on.

Semantic Parsing 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. The CoNaLa dataset (Yin et al., 2018) contains Python snippets and natural language—there are interesting things to do with this, but just trying to map between language and code directly does not work well. There also exist a plethora of datasets for the text-to-SQL task (Finegan-Dollak et al., 2018), which you can investigate.

Multilingual settings 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.

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.

QA / Machine Reading A plethora of question-answering datasets have been released recently: SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), RACE (Lai et al., 2017), WikiHop (Welbl et al., 2017). Many of these datasets are hard to improve on with fully end-to-end neural network approaches (you need large BERT models to push the state-of-the-art). However, recent new datasets are always emerging (Dasigi et al., 2019; Lin et al., 2019). You can also feel free to tackle a subset of examples in a dataset or try out an interesting technique other than end-to-end neural networks. Building faster QA models is also potentially an interesting direction.

Miscellaneous (1) BERT has a "next sentence prediction" module. This has not been extensively investigated. Suppose you have two sentences s_1 and s_2. Given a bunch of other sentences $s'_1, \ldots, s'_n$, if NSP(s_1,s'_i) $\approx$ NSP(s_2,s'_i) for all i, does that imply that s_1 and s_2 are paraphrases or anything interesting about them? (2) There are plenty of interesting things you can do with masked language models, such as testing them to see what explicit knowledge they contain (Petroni et al., 2019). Think about what you might be able to do with these!

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 should **not** focus on new pretraining techniques for language models. Such experiments are too large-scale to feasibly execute even if you have access to significant other compute resources. Fine-tuning BERT-Base on a dataset can often be done effectively with more limited resources, but will still typically require GPUs. Moreover, the tasks of 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.

Computational Resources Available

This course has an allocation on the Maverick2 cluster on TACC. The class has 2000 SUs across 50 students, meaning that each student gets roughly 40 node-hours on this resource (though in practice you can use more since not every student will be using TACC). Try to reserve this for when your model is working and you need to run full-scale experiments.

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

Pradeep Dasigi, Nelson F. Liu, Ana Marasovic, Noah A. Smith, and Matt Gardner. 2019. QUOREF: A Reading Comprehension Dataset with Questions Requiring Coreferential Reasoning. In *arXiv*.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named Entity Disambiguation for Noisy Text. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL)*.

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)*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning Over Paragraph Effects in Situations. In *arXiv*.

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)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *Proceedings of AAAI*.

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)*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing Datasets for Multi-hop Reading Comprehension Across Documents. In *arXiv*.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to Mine Aligned Code and Natural Language Pairs from Stack Overflow. In *International Conference on Mining Software Repositories*, MSR, pages 476–486.