CS395T Final Project

Proposal Due: November 9, 9:30am
Presentation: December 5/7, 9:30am
Final Report Due: December 15, 11:59pm

Collaboration You are free to work on this project either individually or in teams of two. Both partners should contribute equally to the submission, and both 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 (and even encouraged 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 structure (e.g., genomics data, time-series data, etc.). Investigating feedforward neural network architectures on the UCI repository 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 30 points total (50% of your course grade). The deliverables are as follows.

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, 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: 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 (20 points) The primary deliverable is a paper written in the style of an ACL/NIPS/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).

For people working individually, this might be around 4 pages excluding references; for teams of two, this might be closer to 8 pages excluding references. However, don't treat these as hard page 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. 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 projects 1 through 3) 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 (5 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 (5 points)**: Is the idea technically sound? Do you describe what seems like a convincing implementation? Is the experimental design correct?
- **Results/Analysis (5 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.
- **Substance (5 points)**: Is the scope of the report and the work done appropriate for a final project? Is the project sufficiently sophisticated and meaty, or does it seem incomplete?

Final Presentation (5 points) During the last week of class, everyone will give a 5-minute presentation on their project. This presentation should state the problem, describe the methodology used, and give highlights of the results. Because the projects won't have been due yet, these results may be preliminary. Teams will be assigned a presentation date randomly at the time the proposal is due.
Grading: 5 points for giving a presentation.

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? Doing a project of this latter form can be tricky because it can sometimes be hard to find the right method of rigorously characterizing the data—this can be as challenging as building a strong system!

Your project should be *novel work*: you shouldn't set out to redo what others have done. However, implementing someone else's model or downloading and running an existing model are worthy first steps. 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 three 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 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. If you want to explore parsing more, a good parser to start from might be the parser of Qi and Manning (2017), which is a neural network version of the parser from class with several different transition systems available.

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. The paper gives some strong baselines, but you might be able to improve on them!

Summarization Several recent papers have addressed summarization with neural networks. Some work has addressed what could more properly be called sentence compression (Chopra et al., 2016), while other work has tackled full summarization using either a dataset of CNN/Daily Mail highlights (Cheng and Lapata, 2016) or a dataset based on New York Times articles (Paulus et al., 2017). Methods for summarization are somewhat similar to those used for machine translation (encoder-decoder models producing summaries from documents), but the inputs and outputs are much longer and more complex than those for machine translation.

Dialogue A recent dataset and dialogue framework released by Facebook called `parl.ai` focuses on several dialogue subtasks for building a goal-oriented multi-turn dialogue system (Bordes et al., 2017). Interpreting sentences and forming API calls looks like parsing, but dialogue state must also be tracked across utterances—there are several challenges here.

QA / Machine Reading A plethora of question-answering datasets have been released recently including SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), RACE (Lai et al., 2017), WikiHop (Welbl et al., 2017). Each of these datasets has different properties. SQuAD in particular has seen a lot of work recently (see the online leaderboard), but is a relatively straightforward problem. The other datasets are more complex in ways that defy traditional NLP techniques. Building models to work well broadly on these datasets may be too ambitious for a course project, but handling a subset of examples may be possible.

Machine Translation A good resource to explore for machine translation is the Europarl corpus, which contains parallel data for many pairs of European languages. While this dataset is a bit artificial, it's publicly available and easy to deal with. Full-scale machine translation models are computationally intensive to train and evaluate, so you might investigate low-resource machine translation settings to make your life a bit easier.

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!

Computational Resources Available

This course has an allocation on TACC. Each student gets roughly 40 hours of compute time (1000 SUs) on large compute nodes. 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

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning End-to-End Goal-Oriented Dialog. In *arXiv*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural Summarization by Extracting Sentences and Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

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)*.

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)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A Deep Reinforced Model for Abstractive Summarization. In *arXiv*.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A Novel Transition System for Dependency Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (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)*.

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