CS224 Final Programming Project Guidelines

Abstract due Mon 20 May, 5 p.m.

Final Programming Project due Mon 3 June, 5 p.m. (late days may be used)

Three minute class presentation Wed 5 June, in class

This project is an opportunity for you to work for longer on a bigger NLP system in an area of your choice! The projects will be judged on creativity in defining the problem to be investigated, and the methods used, thoroughness in considering and justifying your design decisions, and quality of your write-up, including your testing of the system and reporting of results. You will not be penalized if your system performs poorly, providing your initial design decisions weren't obviously unjustifiable, and you have made reasonable attempts to analyze why it failed, and to examine how the system might be improved.

The final project can be a group project. Indeed, we'd strongly encourage you to work as a group, so you can attempt something larger and more interesting. However, the amount of work should be appropriately scaled to the size of the group, and *you should include a brief statement on the responsibilities of different members of the team.* Team members will normally get the same grade, but we reserve the right to differentiate in egregious cases. In general we would like groups size is 2 or 3 – if you're considering a bigger group, you need to talk to us and convince us that a group of greater than 3 is manageable given the inherent parallelizability of the task, and the time available to organize and implement the system. Solo projects are allowed.

You are free (and, where appropriate, encouraged) to make use of existing code and systems as part of your project, but you should make sure their use is properly acknowledged, and make clear what additional value your project is adding.

The first deadline is to submit a project abstract. We've given you a deadline of two weeks, but you're encouraged to submit it earlier if you've worked out what you are doing, or just if you want feedback sooner. This will not be graded, but is there to encourage you to get into a team and organized early, and work out what focused project you are working on. It is also a chance for dialog between the instructors and the team – we can give you extra references, and also information on whether we think the scope of the project is too small or two big. In addition, feel free to make an appointment to talk to Chris or Andrea about possible projects. The project abstract should fit on one page and should be organized around these 3 sections:

- Problem to be investigated
- Approach
- Plan of work

It can either be sent by handed in on paper or sent via email to `cs224n-staff@lists`. Plain ASCII is fine.

1 Data

A fair amount of natural language data of various sorts is available at Stanford. This includes collections from major publishers such as the Linguistic Data Consortium (http://www.ldc.upenn.edu/), and some smaller collections, such as text categorization and information extraction training and test sets. The biggest amount of this data is in English, but there is also some in major foreign languages, and some parallel text. You can access much of this data under AFS at /afs/ir/data/linguistic-data/, but there are other collections, such as most speech data, which are not online, so do ask or look around at catalogs, such as the LDCs to see what else exists. Again, please note that nearly all of this data is licensed to Stanford, and do not copy it to other machines or give it to other people. There is also a lot of data on the web (books and ezines, as well as web pages). If you could use some resources such as tagged or parsed text, or aligned multilingual materials, and you are not sure what there is, let us know. Preferably as soon as possible. You can find some links to corpora and existing tools at:

http://nlp.stanford.edu/links/statnlp.html
http://nlp.stanford.edu/fsnlp/

Two sources that I'd recommended in previous years for web data sets and some interesting projects are the following. They're still both on the web, but neither seems to be being actively maintained at the moment. . . .

Tom Mitchell's group: http://www.cs.cmu.edu/~TextLearning/

Ion Muslea's RISE archive: http://www.isi.edu/ muslea/RISE/

2 Size and Grading

Always a difficult one to define! But roughly you should be aiming for each member of the team to do about twice as much work as on the other programming project. You should aim to do something that is interesting, not just an exercise in programming. This may only be a fairly modest extension of an existing technique, but there should be a clear focus in terms of what you hope to achieve, or hope to show.

Your project write-up should be adequate, but doesn't need to scale linearly in size. A three person project may well find that a 10 page write-up is quite sufficient. As usual, the quality of your write-up is very important. It's hard to define exactly what the write-up should cover, because it depends on the project, but generally, we're looking for:

- Investigation of a research question. There should be a clear question or application, and hypotheses about the answer or a good approach.
- A clear and complete discussion of the algorithms/method used, and a high level description of the implementation.
- A discussion of the testing you did and results you obtained.
- A discussion of alternatives or things you tried to improve performance, and how they fared. While it is okay to try things just to see what happens, this in part includes thoughtful revisions, and argument as to why things were good to try, rather than just making random changes.

- Clear results showing the performance of the system
- Brief but adequate discussion of related work in the literature (as in an academic conference paper)
- Clean, intelligible code.
- Qualitative discussion on linguistic assumptions of the model and their validity.

3 Submission

The project should be electronically submitted be by the date shown in the title (you may take late days). We would also like to get *two* copies of your final report on paper (to facilitate our grading). Paper materials may be submitted in the usual way (to Chris's office, in class, or to Andrea). We would like you to include with your electronic submission a web-readable (HTML, PDF) version of your report, which we will post so others can look at it, and see the kinds of projects people did.

To submit your program, put all the needed files in one directory on a Sweet Hall machine (or, possibly, another machine with AFS on it and with you being correctly authenticated) and type:

```
/afs/ir/class/cs224n/bin/submit-fp
```

If you need to resubmit it type

```
/afs/ir/class/cs224n/bin/submit-fp -replace
```

You should make sure that you include the source code for your programs, a Makefile that will build them, and an electronic copy of the report. If you have any special requirements that make this impractical (e.g., dependencies on large external packages), let us know.

4 Project ideas

Here are some possible project ideas. You are welcome to think up your own, providing they have some reasonable relationship to NLP and topics discussed in this course. If you think it might be controversial, discuss it with one of the staff *before* the final project abstract due date. A good source of ideas could be recent NLP conferences, or just finding a problem that could use some language processing or understanding.

1. Construct a text classification systems, which categorizes texts into useful categories. See M&S exercise 16.3 or 16.5. This should in some way go beyond what we did for WSD systems, e.g., perhaps by handling hierarchical classification.
2. Construct a text segmentation system, which divides a text into pieces on certain topics. Look at section 15.5, and perhaps exercise 15.8.
3. A system for some task in natural language clustering, such as finding related web pages, or learning part of speech categories from raw data, for which you might look at (Schütze 1993, Schütze 1995). Or, one could attempt to use clusters to improve the quality of a language model, or predicting what objects a verb takes.

4. An information extraction system. This could aim to extract relations (frames) from a certain type of text (newspaper reports, biology articles, ...). One concrete example that would probably need to combine use of formatting with other IE techniques: try to design a system that takes email conference announcements, and extracts a frame of conference title, place, conference date, submission date, URL for further information. There are some data sets in the IE directory in AFS.

5. Write an information extraction system that extracts some type of content out of web pages – perhaps products and prices, or products and an overall evaluation of how good they are in reviews. Such systems commonly make more use of markup. Consider using machine learning methods as well as hand-built detectors and classifiers. As a second example, you might start with the corporate information data set from Tom Mitchell's group, and try to extract information about companies. As a third example, look at some of the wrapper generation data sets that can be found off Ion Muslea's page: there are a number of interesting data sets there for restaurants, etc.

6. Write a text summarization system. This could take press releases, or newswire reports, and summarize them down to a few sentences. A lot of early – and continuing – work in text summarization just selects whole sentences that best summarize a document (Kupiec et al. 1995, Salton et al. 1994, Goldstein et al. 1999). Some interesting recent work (Jing and McKeown 1999) attempts to actually take parts of sentences in a sensible fashion. There's a bibliography of work at:

   http://www.ics.mq.edu.au/~swan/summarization/bibliography.htm

7. Produce a simple natural language question and answer system. For example one might produce a system that can respond to questions about how to get form one place to another around the Stanford campus. Or one that can answer questions about geography, like the classic CHAT-80 system. Or one that can do transactions with the users, giving information about movies, and booking tickets, for instance. This would involve building both a parser, a simple knowledge representation system, and a generator that can express knowledge as English. It'd be good if the system could deal intelligently with some ambiguity and underspecification.

8. Write (part of) a text-to-speech system. Implement a method for converting text to phoneme sequences, which can be used as input to a speech synthesizer. A good free speech synthesis system is the Edinburgh Festival speech synthesis system:

   http://www.cstr.ed.ac.uk/projects/festival/

9. Build (part of) a simple statistical machine translation system, and evaluate its effectiveness. The starting point is doing word for word translation based on finding words that correspond in bilingual texts. See M&S chapter 13, or (Melamed 1997). A reasonable small target would simply to build a model of word alignments (i.e., an automatically learned bilingual dictionary) rather than a complete translation system.

10. Build a statistical parser. It would perhaps be best to build one that just works with unlexicalized categories – which usefully reduces the scale of the task. There are still a number of useful hypotheses you could experiment with about how different kinds of

conditioning information could be employed to improving parsing accuracy. You could look at exercise 12.6 or 12.9 or other recent work like (Collins 1996, Collins 1997, Charniak 1996, Charniak 1997a, Charniak 1997b, Manning and Carpenter 1997, Eisner 1996).

11. Investigate structural learning using a probabilistic context-free grammar or some extension thereof of an unannotated corpus – perhaps a small artificially generated one. Can one get useful results? (Some results suggest usually not (Charniak 1993), but one might consider grammatical restrictions on the forms of grammars. Some early work in this direction can be found in (Briscoe and Waegner 1992). We've done some recent more successful work in grammar induction (Klein and Manning 2001)).)

12. Design and implement a part-of-speech tagger.

13. Consider, implement, and test measures of semantic similarity for trees – see M&S exercise 8.22.

14. Conduct a fuller investigation of some aspect of word sense disambiguation, perhaps focussing on comparing alternate methods – there are already some comparisons, such as (Mooney 1996) – or on issues in the non-discreteness of word senses. Perhaps word usages could be placed in a sense space, and a word represented as a density function of senses?

15. Investigate using topics or context for improving a language model (that could be used for OCR or speech recognition). An early piece of work on this was (Lau et al. 1993).

16. Investigate collocation finding (Chapter 5) – perhaps focussing on discontinuous collocations, ones longer than two words, or issues in better ways of identifying good collocates.

17. Develop a good sentence boundary detection system: see M&S section 4.2.4, or (Mikheev 2000).

18. Explore extending statistical and corpus-based methods to areas of semantics and discourse. There has been quite a bit of work in this area in recent years. See recent Association for Computational Linguistics proceedings (Math&CS library). For example one can try to predict the nature of successive turns in a dialogue (question, clarification, ...).

19. Try building a system that learns hyperonym hierarchies (ISA links) automatically from text corpora. Or PART-OF hierarchies. Eugene Charniak and students have some recent papers on this topic. See:

    http://www.cs.brown.edu/people/ec/
    http://www.cs.brown.edu/people/sc/

    Or (Riloff and Jones 1999).

20. Write a system to correctly choose the order among multiple adjectives modifying a noun (so it knows that *an old green leather couch* sounds okay, but *??a leather green old couch* sounds funny.

21. ... (use your creativity!)

Bibliography

Briscoe, T., and N. Waegner. 1992. Robust stochastic parsing using the inside-outside algorithm. In *Proceedings of the AAAI '92 Workshop on Probabilistically-Based Natural Language Processing Techniques*, 39–53. AAAI. Revised version at http://www.cl.cam.ac.uk/Research/Papers/.

Charniak, E. 1993. *Statistical Language Learning.* Cambridge, MA: MIT Press.

Charniak, E. 1996. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI '96)*, 1031–1036.

Charniak, E. 1997a. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI '97)*, 598–603.

Charniak, E. 1997b. Statistical techniques for natural language parsing. *AI Magazine* 33–43.

Collins, M. J. 1996. A new statistical parser based on bigram lexical dependencies. In *ACL 34*, 184–191.

Collins, M. J. 1997. Three generative, lexicalised models for statistical parsing. In *ACL 35/EACL 8*, 16–23.

Eisner, J. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 16*, 340–345.

Goldstein, J., M. Kantrowitz, V. Mittal, and J. Carbonell. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *SIGIR '99*, 121–128.

Jing, H., and K. McKeown. 1999. The decomposition of human-written summary sentences. In *SIGIR '99*, 129–136.

Klein, D., and C. D. Manning. 2001. Natural language grammar induction using a constituent-context model. In *Advances in Neural Information Processing Systems*, Vol. 14. MIT Press.

Kupiec, J., J. Pedersen, and F. Chen. 1995. A trainable document summarizer. In *SIGIR '95*, 68–73.

Lau, R., R. Rosenfeld, and S. Roukos. 1993. Adaptive language modeling using the maximum entropy principle. In *Proceedings of the Human Language Technology Workshop*, 108–113. ARPA.

Manning, C. D., and B. Carpenter. 1997. Probabilistic parsing using left corner language models. In *Proceedings of the Fifth International Workshop on Parsing Technologies (IWPT-97)*, 147–158, MIT.

Melamed, I. D. 1997. A portable algorithm for mapping bitext correspondence. In *ACL 35/EACL 8*, 305–312.

Mikheev, A. 2000. Tagging sentence boundaries. In *NAACL 1*, 264–271.

Mooney, R. J. 1996. Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In *EMNLP 1*, 82–91.

Riloff, E., and R. Jones. 1999. Learning dictionaries for information extraction using multi-level bootstrapping. In *AAAI 1999*.

Salton, G., J. Allan, C. Buckley, and A. Singhal. 1994. Automatic analysis, theme generation and summarization of machine-readable texts. *Science* 264:1421–1426.

Schütze, H. 1993. Part-of-speech induction from scratch. In *ACL 31*, 251–258.

Schütze, H. 1995. Distributional part-of-speech tagging. In *EACL 7*, 141–148.