CS224N/Ling 280 Final Programming Project Guidelines

Project Abstract due Wed 7 May, midnight.

Final Programming Project due Wed 4 June, midnight.
Brief "elevator pitch" project presentation in CS224N exam slot Mon 9 June morning

This project is an opportunity for you to work on an NLP system in an area of your choice! The projects will be judged on creativity in defining the problem to be investigated, the methods used, thoroughness in considering and justifying your design decisions, and quality of your write-up, including your testing of the system, error and success analysis, 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. The amount of work should be appropriately scaled to the size of the group (though the expected scaling is sublinear), 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 group sizes of 2 or 3 – if you're considering a bigger group, you *must* 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. For the final project, group size is considered in the grading, but even someone working alone has to complete a good project to get a good grade.

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.

1 Project Abstract

The first deadline is to submit a *project abstract.* This will not be graded, but is there to encourage you to get 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. You can tell us what you plan to do, anything you have achieved so far, and what you hope to achieve in the rest of the quarter. We can give you extra references, and also information on whether we think the scope of the project is too small or two big. So please do think about where you are and have something focused and concrete ready to submit. This milestone is to put some uniform structure into the process, but beyond that, we *really encourage* you to stop by one of our office hours to discuss projects in person. That allows longer and more productive discussion of projects. Talking about project plans is a particularly good place to get useful feedback and information from the course staff. Just due to greater experience or a different viewpoint, this can often help a lot. The project progress report should fit on one page and should be organized around these 4 sections:

- Problem being investigated
- Approach/plan

- Any achievements so far and/or relevant references found
- Plan of work for the remainder of the quarter

It can either be handed in on paper or sent via *plain text* email to `cs224n-spr0708-staff@lists`. *Please put your email address(es) on the final project proposal* – this will make it easier for us to send feedback.

2 Data

A quite large 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 data in major foreign languages (Chinese, German, French, Arabic, ...), 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 LDC's to see what else exists. The site that discusses corpora available at Stanford is at:

`http://www.stanford.edu/dept/linguistics/corpora/`

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 (free corpora and bake-off data, books, blogs, and 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/

3 Size and Grading

Always a difficult one to define! But roughly you should be aiming for each member of the team to do at least as much work as on one of the homeworks. You should aim to do something that is small but interesting (i.e., 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. It's perfectly okay to extend something you did in an earlier homework.

Your project write-up should be adequate, but doesn't need to scale linearly in size. One person might want to write 6 pages. A three person project may well find that a 10 page write-up is quite sufficient. Think of the write-up as something like a conference paper, focussed on research questions and achievements, though you may want to include a bit more detail on methods used, examples, etc. You could even look at example computational linguistics conference papers: see the site at http://aclweb.org/anthology-new/. 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. (Make use of tables, graphs, etc.!)
- 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.

4 Submission

The project should be electronically submitted by the date shown in the title (you may take up to 5 late days). We want to receive ***three 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 a TA). We would be pleased if you could include with your electronic submission a web-readable (HTML, PDF, DOC) 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
```

You should make sure that you include the source code for your programs, a Makefile or build.xml 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.

Finally, we will have *brief* presentations of projects during the scheduled exam period for the class: this is a good opportunity to learn all the exciting things people have been doing!

5 Project ideas

Here are some possible project ideas. You are *encouraged* to think up your own, providing they have some reasonable relationship to NLP and topics discussed in this course. The best projects are commonly interesting and nifty projects that students think up for themselves rather than the tired old list of ideas that your professor writes down. One approach is to design some system that incorporates NLP into an application. If you think your project idea 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. Many, many NLP conference papers are available online at:

```
http://acl.ldc.upenn.edu/
```

Question Answering

1. One part of typical open domain question answering systems is a *question classifier* that determines the semantic type of the desired answer (a person name, height, temperature, etc.). Attempt to build such a system. There's a paper on this and a nice data set available at:

   ```
   http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/
   ```

2. 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 from 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 users, giving information about movies, and booking tickets, for instance. This would involve building or using 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, or unknown words.

Information Extraction

1. Write 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.

2. Write an information extraction system that extracts some type of content out of web pages – perhaps product names and prices (e.g., Google's Froogle attempts to do this, though in practice they mainly show results from structured data feeds), 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.

Machine Translation

1. Improve your existing machine translation model components!! Probably the most exciting thing to do would be to build or utilize a decoder, which you could couple with your existing language model and word alignment models to provide a complete statistical MT system (and then we'll really see how good it is!). A key reference on statistical decoders is Germann et al.'s:

   `http://www.isi.edu/natural-language/projects/rewrite/decoder.pdf`

   But you can find much more detail on how to build a uniform cost decoder in Mike Jahr's thesis (he was a symsys student at Stanford, who worked a couple of summers at ISI, and is now doing MT at Google):

   `http://dbpubs.stanford.edu:8090/pub/2001-59`

   Rather than building your own decoder, an alternative would be to use an existing decoder. Recently, Marian Olteanu at UT Dallas released an open source Java decoder:

   `http://www.phramer.org`

   You could work out how to hook our language models and word alignment models with it.

2. Another way you could improve on your existing MT components would be to investigate better alignment models. Two obvious things to try are to build a Model 3 alignment model, or else to build an HMM-based alignment model. HMM-based alignment models are the most popular word-based model not in the classic set of IBM models. See:

Franz Josef Och, Hermann Ney: Improved Statistical Alignment Models. ACL 2000.
http://acl.ldc.upenn.edu//P/P00/P00-1056.pdf
Kristina Toutanova, H. Tolga Ilhan and Christopher D. Manning, Extensions to HMM-based StatisticalWord Alignment Models.
http://www.stanford.edu/~krist/papers/hmmalign.pdf

3. A third thing you could do is look at a phrase-based alignment model, which (as mentioned in class) is currently seen as the most promising approach. There are datasets and old competition results at:

   http://www.statmt.org/wpt05/mt-shared-task/
   http://www.statmt.org/wmt06/shared-task/
   http://www.statmt.org/wmt07/shared-task.html

   As noted at http://www.statmt.org/wmt06/shared-task/baseline.html, there is pretty much a "standard tool chain" for baseline statistical MT, using the Giza++ alignment model toolkit, a language model builder (such as Carmel, or the SRI language modeling toolkit), and a decoder such as Pharoah or Phramer. You could also work from this baseline.

4. Tired of working on major world languages? Now here's a *really* interesting challenge! See if you can build a system for English-Inuktitut (Canadian "Eskimo") word alignment or machine translation. You can find data here:

   http://www.cs.unt.edu/~rada/wpt05/

   See also the earlier workshop at:

   http://www.cs.unt.edu/~rada/wpt/

Parsing and POS tagging

1. Extend your statistical parser. You should probably stick with either unlexicalized categories or just build a word-dependency parser. There isn't time to build a full lexicalized parser. 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 FSNLP exercise 12.6 or 12.9 or other recent work including (Collins 1996, Collins 1997, Charniak 1996, Charniak 1997a, Charniak 1997b, Manning and Carpenter 1997, Eisner 1996, Klein and Manning 2003, McDonald et al. 2005, Eisner and Smith 2005).

2. Examine statistical parsing of data in a different language: we now have parsed corpora (of some sort and size) for several languages, including Czech, German, Arabic, and Chinese, Danish. Either adapt your parser to work with another language or consider adapting or improving ours:

   http://nlp.stanford.edu/downloads/lex-parser.shtml

   to a new language. An example of trying to do this for Chinese is (Levy and Manning 2003).

3. We've got a Java statistical parser, available with source, at:

   http://nlp.stanford.edu/downloads/lex-parser.shtml

   It's not bad, but not the world's best. It'd be good if it was better! Some ways that seem hopeful for bettering it include:

   · Improving the dependency grammar model used in the factored parser

- The PCFG part currently just uses MLE of rules. One *should* be able to get better performance by smoothing rule probabilities, though it has seemed in practice a bit difficult to get parsing accuracy gains, and difficult not to slow it down enormously by trying.
- Providing a 3 stage backoff in the lexicon and dependency grammar from word to fine tag to coarse tag, whereas at the moment there is only a 2 stage backoff.

4. Write a part-of-speech tagger. There is already quite a bit of support in the `cs224n` code for this, but crucial parts still need to be supplied. Another good possibility is to look a POS tagging a language other than English.

Grammar Induction

1. 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 2002, Klein and Manning 2004), or see also Smith and Eisner (2005). The 2002 system has been successfully reimplented by students as a class project already!

Clustering and Classification

1. Write 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, Clark 2003). Or, one could attempt to use clusters to improve the quality of a language model, or predicting what objects a verb takes.

2. Construct a text classification systems, which categorizes texts into useful categories. See M&S exercise 16.3 or 16.5. This should try to do something interesting, e.g., perhaps by handling hierarchical classification. One possible domain of application here is to write an email spam filtering system.

3. Text classification can be directed at many purposes other than topic, and many of the alternatives are more interesting and give more opportunities for using linguistic features. One version that has been explored a bit in recent years is positive vs. negative opinion. But there still aren't systems that work very well, and there are clear challenges to find features that work better. For starters, see (Turney 2002, Pang et al. 2002). The data for the latter system is available from Lillian Lee's web site.

Anaphora resolution

1. Write a system for anaphora resolution: when mentions of a noun or a pronoun refer back to something introduced earlier (preferably doing *that problem* as well as *him*). There are more recent papers, but one good one to look at is (Ge et al. 1998). Some annotated data is available at `http://clg.wlv.ac.uk/resources/corefann.php` .

Text summarization and segmentation

1. 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

2. Construct a text segmentation system, which divides a text into pieces on certain topics. Look at section M&S 15.5, and perhaps exercise 15.8.

The List goes ever on and on, down from the door where it began. Now far ahead the List has gone, and I must follow, if I can

1. Consider, implement, and test measures of semantic similarity for trees – see M&S exercise 8.22.
2. Conduct an investigation of some aspect of word sense disambiguation, perhaps focussing on comparing alternate methods 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? There are lots of resources at:

   `http://www.senseval.org/`
   `http://www.cs.unt.edu/~rada/downloads.html`

3. 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).
4. Investigate collocation finding (M&S, Chapter 5) – perhaps focussing on discontinuous collocations, ones longer than two words, or issues in better ways of identifying good collocates.
5. Develop a good sentence boundary detection system: see M&S section 4.2.4, or (Mikheev 2000).
6. 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 at the UPenn site listed above. For example one can try to predict the nature of successive turns in a dialogue (question, clarification, ...).
7. 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) or `http://ai.stanford.edu/~rion/papers/hypernym_nips04.pdf`.

8. 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.
9. Write a natural language generation system, for data in a certain domain. We didn't cover this topic in class. One good starting point would be: Reiter and Dale (2000).
10. In recent years there has been lots of exciting work in learning lexical semantics from large amounts of text, including locally from Rion Snow:

    `http://ai.stanford.edu/~rion/papers/semtax_acl06.pdf`

    or from other groups:

http://www.patrickpantel.com/cgi-bin/Web/Tools/getfile.pl?type=paper&id=2007/naacl07-01.pdf
http://www.patrickpantel.com/cgi-bin/Web/Tools/getfile.pl?type=paper&id=2002/kdd02.pdf

Such things can make good projects (but beware of trying to do something at a large scale than you can handle computationally!).

11. There are a lot of competitions for grown-ups in the NLP world, where groups compete to build systems that outperform other systems. Many of these are on reasonable size tasks that people could attempt. This framework has the advantage of providing a data set and the results of other systems (providing the competition isn't too recent) – at the cost of perhaps decreasing your creativity in defining your own problem. Here are some other examples you could look at:

    (a) CoNLL Shared Tasks: named entity recognition, semantic predicate argument structure, phrasal chunking etc. See the list at:

        http://cnts.uia.ac.be/signll/shared.html

    (b) Chinese Word Segmentation

        http://www.sighan.org/bakeoff2003/

    (c) DUC: There have been competitions in document summarization, but it's much harder to get the data for those (whole bunch of forms need signing...)

        http://www-nlpir.nist.gov/projects/duc/index.html

    (d) BioMedical Named Entity Recognition and Information extraction tasks

        - http://www.pdg.cnb.uam.es/BioLINK/BioCreative.eval.html (BioCreative)
        - http://research.nii.ac.jp/~collier/workshops/JNLPBA04st.htm Coling2004 BioNLP Named Entity Recognition task (just finished – I entered this with a couple of students)
        - http://medir.ohsu.edu/~genomics/ (TREC Genomics – high overhead to access)
        - CleanEval http://cleaneval.sigwac.org.uk

12. ... Use your own creativity!

13. Look at some of the efforts of past students at:

        http://nlp.stanford.edu/courses/cs224n/

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.

Clark, A. 2003. Combining distributional and morphological information for part of speech induction. In *EACL 2003*.

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.

Eisner, J., and N. A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of the International Workshop on Parsing Technologies (IWPT 2005)*.

Ge, N., J. Hale, and E. Charniak. 1998. A statistical approach to anaphora resolution. In *WVLC 6*, 161–170.

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. 2002. Natural language grammar induction using a constituent-context model. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14*, Vol. 1, 35–42. MIT Press.

Klein, D., and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL 41*, 423–430.

Klein, D., and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL 42*, 479–486.

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.

Levy, R., and C. D. Manning. 2003. Is it harder to parse Chinese or the Chinese Treebank? In *ACL 41*, 439–446.

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.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *ACL 43*.

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

Pang, B., L. Lee, and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP 2002*, 79–86.

Reiter, E., and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge: Cambridge University Press.

Riloff, E., and R. Jones. 1999. Learning dictionaries for information extraction using multi-level boot-strapping. 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.

Smith, N. A., and J. Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *Working Notes of the International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Grammatical Inference Applications*.

Turney, P. D. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL 2002*, 417–424.