11-712: NLP Lab

Noah A. Smith

Spring 2013

The focus of this lab is natural language morphology. To successfully complete the laboratory exercise, you will design, implement, evaluate, document, and openly release a finite-state morphological analyzer for a language of your choice.

1 Background and Goals

Morphological analysis is an essential part of NLP for many languages. Despite major advances in formal machinery and descriptive linguistics relevant for building morphological models, morphological analyzers are only available for a few languages. Many of them are proprietary, not open-source, and/or not built for iterative improvement and adaptation.

The goal of this lab is to give you hands-on experience developing and documenting an open-source NLP tool. It is our hope that you will build a tool with good enough performance to be useful to others, and that you will design it in such a way that it can be extended and improved by others.

Some useful readings:

- Roche and Schabes (1997), chapters 1–4, 11
- Beesley and Kartunnen (2003)
- Roark and Sproat (2007), chapters 1–5
- Çöltekin (2010)
- Pirinen (2011)
- Lindén et al. (2013)
- FOMA tutorial: `https://code.google.com/p/foma/wiki/MorphologicalAnalysisTutorial`
- Apertium wiki: `http://wiki.apertium.org/wiki/Category:Morphological_analysers`

Some useful tools:

- GitHub (`http://github.com`): version control and software publishing platform
- FOMA (`http://code.google.com/p/foma`): open-source library for developing morphology FSTs
- HFST (`http://www.ling.helsinki.fi/kieliteknologia/tutkimus/hfst`): another open-source library for developing morphology FSTs

- OpenFST (`http://www.openfst.org`): useful for dealing with *weighted* transducers (which you are unlikely to need in this lab)
- XFST (`http://www.fsmbook.com`): Xerox tools, not yet open-source (FOMA may essentially make this obsolete)
- FSM Library (`http://www2.research.att.com/~fsmtools/fsm`): AT&T's tools (made obsolete by OpenFST)

2 Schedule

The lab is tightly structured across 15 weeks. You will be writing documentation from the very beginning. There are deadlines every one or two weeks. The amount of work each week is relatively small, but you must complete each milestone on schedule.

1. By 1/18: Select a language.[1] If possible, you should identify a native speaker to whom you can turn for judgments. Ideally this person will have enough linguistics training to have conversations with you about stems, inflections, parts of speech, and so on. This person will need to agree to spend roughly two hours judging analysis output with you, three times during the semester. If you cannot find a native speaker, try to find someone who knows the language (other than yourself). If this is a major difficulty, discuss alternatives with the instructor. Also, create an open-source project on GitHub where you will openly develop your project and write your report. Write part 1 of the report.
2. By 1/25: Identify and review past work on this language's morphology. You must choose a language with a writing system in which tokenization into words will be easy. This includes linguistics papers, computational modeling papers, and system descriptions. To the greatest extent possible, you will want to *incorporate* previous work. Write part 2 of the report.
3. By 2/1: Identify existing resources that might be useful in building your analyzer. Open-source lexicons, corpora, and reference grammars are worth looking for. You will need two development corpora of at least 1,000 words, which you should not directly inspect, and a third corpus of at least 10,000 words. Call these A, B, and C. Write part 3 of your report.
4. By 2/8: Catalog the attested phenomena in the language, based on your literature search, the resources you've identified, and a meeting with your informant. Don't forget about spelling rules, since you need to handle text. Prioritize this list. Write part 4 of your report.
5. By 2/15: Based on complete set of phenomena you would eventually cover, organize the problem into modules. For example, you might include a lexicon of open-class stems for each major part of speech, FSTs for inflections on different parts of speech, spelling rules, and so on. Some modules might be developed in stages, so identify for each what the first version, second version, and so on will need to be capable of handling. Prioritize these, based on the phenomena you need to handle. Implement your first module: a "guesser" that gives a sensible default analysis for any input (and marks it explicitly as a "guess"). Write part 5 of your report.
6. 2/16–3/1: Based on your prioritized list, first round of development.

[1] Because we want you to produce a tool that is *useful*, it is not recommended that you choose a language for which sophisticated tools are already available. But this is not a hard constraint.

7. By 3/8: First round of evaluation. For every word in corpus A, run it through your analyzer. Organize the output in a way that will make it easier for your informant to give feedback. Come up with a performance measure appropriate to the language and dataset (probably something like precision and recall per word form). Write part 6 of your report.

8. By 3/15: Write up the lessons you learned and any revisions to the design of your analyzer and prioritized list (part 7 of your report) .

9. 3/16–3/29: Continue development, based on the revised plan.

10. By 4/5: Repeat the evaluation, this time with corpus B. Write part 8 of your report.

11. By 4/6–4/25: Continued development, based on the outcomes of the second evaluation. Ensure that the model runs on all the words in corpus C. Write up part 9 of your report.

12. By 4/26: Upload the words and their analyses for inspection by potential users of your tool. Write up the future work (part 10 of your report), based on the prioritized items you didn't get to. Turn in your report and release version 1.0 of your tool.

3 Report

Your report should be written in LaTeX, using the template provided at `http://www.cs.cmu.edu/~nasmith/NLPLab/report-template.tex`. The outline for your report is fixed:

1. Basic information about the language
2. Past work on morphology for this language
3. Available resources, including your corpora
4. Survey of phenomena
5. Initial design
6. System analysis on corpus A
7. Lessons learned and revised design
8. System analysis on corpus B
9. Final revisions
10. Future work

You must write each portion of the report on schedule and maintain the report in your GitHub repository.

References

Kenneth R. Beesley and Lauri Kartunnen. *Finite State Morphology.* CSLI Publications, 2003.

Çağrı Çöltekin. A freely available morphological analyzer for turkish. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, 2010. URL `http://www.lrec-conf.org/proceedings/lrec2010/pdf/109_Paper.pdf`.

Krister Lindén, Erik Axelson, Senka Drobac, Sam Hardwick, Miikka Silfverberg, and Tommi Pirinen. Using HFST for creating computational linguistic applications. In Adam Przepirkowski, Maciej Piasecki, Krzysztof Jassem, and Piotr Fuglewicz, editors, *Computational Linguistics*, volume 458 of *Studies in Computational Intelligence*, pages 3–25. Springer Berlin / Heidelberg, 2013. URL `http://dx.doi.org/10.1007/978-3-642-34399-5_1`.

Tommi Pirinen. Modularisation of finnish finite-state language description—towards wide collaboration in open source development of a morphological analyser. In *Proceedings of the 18th Nordic Conference of Computational Linguistics*, 2011. URL `http://dspace.utlib.ee/dspace/bitstream/handle/10062/17358/0Pirinen_58.pdf`.

Brian Roark and Richard Sproat. *Computational Approaches to Morphology and Syntax.* Oxford University Press, 2007.

Emmanuel Roche and Yves Schabes, editors. *Finite-State Language Processing.* MIT Press, 1997.