bllipparser Documentation

Release latest

September 03, 2015

Contents

CHAPTER 1

Overview

BLLIP Parser is a statistical natural language parser including a generative constituent parser (`first-stage`) and discriminative maximum entropy reranker (`second-stage`). The latest version can be found on GitHub. This document describes basic usage of the command line interface and describes how to build and run the reranking parser. There are now Python and Java interfaces as well. The Python interface is described in README-python.rst.

CHAPTER 2

Compiling the parser

1. *(optional)* For optimal speed, you may want to define `$GCCFLAGS` specifically for your machine. However, this step can be safely skipped as the defaults are usually fine. With `csh` or `tcsh`, try something like:

```
shell> setenv GCCFLAGS "-march=pentium4 -mfpmath=sse -msse2 -mmmx"
```

   or:

```
shell> setenv GCCFLAGS "-march=opteron -m64"
```

2. Build the parser with:

```
shell> make
```

   - Sidenote on compiling on OS X

     OS X uses the `clang` compiler by default which cannot currently compile the parser. Try setting this environment variable before building to change the default C++ compiler:

```
shell> setenv CXX g++
```

     Recent versions of OS X may have additional issues. See issues 19 and 13 for more information.

CHAPTER 3

Obtaining parser models

The GitHub repository includes parsing and reranker models, though these are mostly around for historical purposes. See BLLIP Parser models for information about obtaining newer and more accurate parsing models.

CHAPTER 4

Running the parser

After it has been built, the parser can be run with:

```
shell> parse.sh <sourcefile.txt>
```

For example:

```
shell> parse.sh sample-text/sample-data.txt
```

The input text must be pre-sentence segmented with each sentence in an <s> tag:

```
<s> Sentence 1 </s>
<s> Sentence 2 </s>
...
```

Note that there needs to be a space before and after the sentence.

The parser distribution currently includes a basic Penn Treebank Wall Street Journal parsing models which `parse.sh` will use by default. The Python interface to the parser includes a mechanism for listing and downloading additional parsing models (some of which are more accurate, depending on what you're parsing).

The script `parse-and-fuse.sh` demonstrates how to run syntactic parse fusion. Fusion can also be run via the Python bindings.

The script `parse-eval.sh` takes a list of treebank files as arguments and extracts the terminal strings from them, runs the two-stage parser on those terminal strings and then evaluates the parsing accuracy with Sparseval. For example, if the Penn Treebank 3 is installed at `/usr/local/data/Penn3/`, the following code evaluates the two-stage parser on section 24:

```
shell> parse-eval.sh /usr/local/data/Penn3/parsed/mrg/wsj/24/wsj*.mrg
```

The `Makefile` will attempt to automatically download and build Sparseval for you if you run `make sparseval`.

For more information on Sparseval see this paper:

```
@inproceedings{roark2006sparseval,
    title={SParseval: Evaluation metrics for parsing speech},
    author={Roark, Brian and Harper, Mary and Charniak, Eugene and
            Dorr, Bonnie and Johnson, Mark and Kahn, Jeremy G and
            Liu, Yang and Ostendorf, Mari and Hale, John and
            Krasnyanskaya, Anna and others},
    booktitle={Proceedings of LREC},
    year={2006}
}
```

We no longer distribute evalb with the parser since it sometimes skips sentences unnecessarily. Sparseval does not have these issues.

CHAPTER 5

More questions?

There is more information about different components of the parser spread across `README` files in this distribution (see below). BLLIP Parser is maintained by David McClosky.

- Usage help: StackOverflow (use `charniak-parser` tag)
- Bug reports and feature requests: GitHub issue tracker
- Twitter: @bllipparser

5.1 Parser details

For details on the running the parser, see first-stage/README.rst. For help retraining the parser, see first-stage/TRAIN/README.rst (also includes some information about the parser model file formats).

5.2 Reranker details

See second-stage/README for an overview. second-stage/README-retrain.rst details how to retrain the reranker. The `second-stage/programs/*/README` files include additional notes about different reranker components.

5.3 Other versions of the parser

We haven't tested these and can't support them, but they may be useful if you're working on other platforms or languages.

- Native Charniak parser for Windows (doesn't need cygwin, no reranker)
- Rutu Mulkar-Mehta's Windows version
- Djame's French branch
- Liang Huang's Forest Reranker (includes forest-dumping extensions)

5.4 References

- Eugene Charniak and Mark Johnson. "Coarse-to-fine n-best parsing and MaxEnt discriminative reranking." Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005.
- Eugene Charniak. "A maximum-entropy-inspired parser." Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference. Association for Computational Linguistics, 2000.

Self-training:

- David McClosky, Eugene Charniak, and Mark Johnson. "Effective Self-Training for Parsing." Proceedings of the Conference on Human Language Technology and North American chapter of the Association for Computational Linguistics (HLT-NAACL 2006), 2006.

Syntactic fusion:

- Do Kook Choe, David McClosky, and Eugene Charniak. "Syntactic Parse Fusion." Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2015), 2015.