Corpus Linguistics (L615)
Automatic POS and Syntactic Annotation

Markus Dickinson
Department of Linguistics, Indiana University
Spring 2013
Where we’re going

Examine POS tagging & parsing

- Focus on getting a few tools working
  - We’ll focus on English today ...

Many taggers/parsers have *pre-built* models; others can be *trained* on annotated data

You’ve already worked with TreeTagger; today, we’ll focus on tools from the Stanford NLP group
Wikis with useful technology information

Places you can get your own information:

- Our very own IU CL wiki, which includes some people’s experiences with various tools
  - [http://cl.indiana.edu/wiki](http://cl.indiana.edu/wiki)
  - Always feel free to add your own experiences to help the next person who wants to use that tool

- ACL wiki & resources
  - ACL software registry: [http://registry.dfki.de/](http://registry.dfki.de/)
The Stanford NLP group has a lot of software tools you might find handy:

- http://nlp.stanford.edu/software/

We will look at the POS tagger & parser separately, but many tools come bundled as part of the CoreNLP package:

Automatic POS Tagging

How do taggers work?

- We talked before about how the general assumption is that local context is sufficient for tagging.

Some examples where this seems to hold true:

- for the **man**: noun or verb?
- we will **man**: noun or verb?
- I can **put**: verb base form or past?
- re-cap **real** **quick**: adjective or adverb?

Bigram or trigram tagging is quite popular.

- Take L545/L645 if you want to know more.
POS taggers

- TnT: http://www.coli.uni-saarland.de/~thorsten/tnt/
  - Trainable; models for German & English

- TreeTagger: http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
  - Trainable; models for English, German, Italian, Dutch, Spanish, Bulgarian, Russian, & French; unix, mac, PC

- Qtag: http://www.english.bham.ac.uk/staff/omason/software/qtag.html
  - Trainable; models for German & English

  - Has a variety of NLP modules

- OpenNLP: http://opennlp.sourceforge.net/
  - Models for English, German, Spanish, & Thai; Has a variety of NLP modules
POS taggers (2)

  - Trainable; integrates different technologies

  - Trainable; models for English, Arabic, Chinese, & German

  - English

Stanford POS tagger

Available at: http://nlp.stanford.edu/software/tagger.shtml

There is a GUI, but most features are available only through command line calls

▶ Note the tutorial on tagging XML files:
  http://www.matthewjockers.net/2008/05/29/
  pos-tagging-xml-with-xgrid-and-the-stanford-log-linear-part-of-speech-
Stanford POS tagger

GUI (stanford-postagger-gui.sh)

Type a sentence to tag:
(Tag this sentence.

Tagged sentence:
(Tag_NN this_DT sentence_NN .)
Stanford POS tagger

Tagging

> ./stanford-postagger.sh \
  models/english-left3words-distsim.tagger \
  sample-input.txt

Loading default properties from tagger models/...
Reading POS tagger model from models/... ... done [1.4 sec].

... A_DT passenger_NN plane_NN has_VBZ crashed_VBN shortly_RB after_IN take-off_NN from_IN Kyrgyzstan_NNP 's_POS capital_NN ,_, Bishkek_NNP ,_, killing_VBG a_DT large_JJ number>NN of_IN those_DT on_IN board>NN ._.
Tagged 72 words at 654.55 words per second.
Stanford POS tagger

Options file (example)

```
models/english-left3words-distsim.tagger.props

## tagger training invoked at Mon May 21 22:54:42 PDT 2012 with arguments:
model = english-left3words-distsim.tagger
arch = left3words,naacl2003unknowns,wordshapes(-1,1),distsim(/u/nlp/data/pos_tags_are_useless/egw4-reut.512.clusters,-1,1),distsimconjunction(/u/nlp/data/pos_tags_are_useless/egw4-reut.512.clusters,-1,1)

wordFunction =

closedClassTags =
closedClassTagThreshold = 40
curWordMinFeatureThresh = 2
debug = false
debugPrefix =
tagSeparator = _
encoding = UTF-8
iterations = 100
lang = english
```

Stanford POS tagger

Setting options

Note that the contents of the stanford-postagger.sh file are as follows:

```bash
java -mx300m -cp 'stanford-postagger.jar:'
edu.stanford.nlp.tagger.maxent.MaxentTagger
-model $1 -textFile $2
```

- `-mx300m` specifies memory usage: you can increase this if need be
- `-model` & `-textFile` specify required parameters
- other options would be, e.g.,:
  - `-verbose`
  - `-tokenize false`
Stanford POS tagger

Training

To train the tagger requires:

1. an options file
   - copy one from models/ & change: 1) models, 2) trainFile, 3) arch
   - also change search option to search = qn

2. data tagged in the way specified by tagSeparator in options file
Stanford POS tagger

Training

Example call (note I’m training on a tiny, automatically-tagged dataset)

> java -mx300m -classpath stanford-postagger.jar \
edu.stanford.nlp.tagger.maxent.MaxentTagger \
-prop training/md.props -model training/md.tagger \
-trainFile sample-output.txt

This creates training/md.tagger & training/md.tagger.props
Parsers attempt to build a tree, based on some grammar
  ▶ They have to process a sentence & tree in a certain order
  ▶ They often disambiguate by using probabilities of rules

Again, take L545/L645 for more details
Constituency Parsers

- **LoPar**: http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/LoPar.html
  - Trainable; models for English & German
- **BitPar**: http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/BitPar.html
  - Trainable; models for English & German
- **Charniak & Johnson parser**: http://www.cs.brown.edu/people/ec/#software
  - Trainable; mainly used for English
Constituency Parsers (2)

- Collins/Bikel parser:
  http://people.csail.mit.edu/mcollins/code.html
  http://www.cis.upenn.edu/~dbikel/software.html
  - Trainable on English, Chinese, and Arabic; designed for Penn Treebank-style annotation

- Stanford parser:
  - Trainable; models for English, German, Chinese, & Arabic; dependencies also available

- Berkeley parser:
  http://code.google.com/p/berkeleyparser/
  - Trainable; models for English, German, and Chinese
Dependency parsing

- MaltParser: [http://w3.msi.vxu.se/~nivre/research/MaltParser.html](http://w3.msi.vxu.se/~nivre/research/MaltParser.html)
  - Trainable; models for Swedish, English, & Chinese
- MSTParser: [http://sourceforge.net/projects/mstparser](http://sourceforge.net/projects/mstparser)
  - Trainable; has some models for English & Portuguese
  - English only

- CCG parsers: [http://groups.inf.ed.ac.uk/ccg/software.html](http://groups.inf.ed.ac.uk/ccg/software.html)
  - Primarily for English, although can be trained on German CCGbank
Stanford parser

There are three basic parsers:

- a high-accuracy unlexicalized PCFG
- a lexicalized dependency parser
- a lexicalized PCFG treebank parser
  - a factored model, with jointly optimization of dependencies & unlexicalized PCFGs

The models are in
edu/stanford/nlp/models/lexparser/ and can be changed in lexparser.sh

- Models for English, Chinese, German, Arabic, & French

There are various extensions on the webpage (e.g., parse visualizer, Python interface)
Stanford parser

GUI (lexparser-gui.sh)

You was butterflying about the personality crisis, you got it while it was hot.
/lexparser.sh data/testsent.txt
Loading parser from serialized file
edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz
... done [1.5 sec].

Parsing file: data/testsent.txt
Parsing [sent. 1 len. 21]: Scores of properties are under extreme fire threat as a huge blaze continues to advance through Sydney ‘s north-western suburbs.
Running the parser

Output (1)

(ROOT
  (S
    (NP (NNS Scores))
    (PP (IN of)
      (NP (NNS properties)))
    (VP (VBP are)
      (PP (IN under)
        (NP (JJ extreme) (NN fire) (NN threat)))
      (SBAR (IN as)
        (S
          (NP (DT a) (JJ huge) (NN blaze))
          (VP (VBZ continues)
            (S
              (VP (TO to)
                (VP (VB advance)
                  (PP (IN through)
                    (NP
                      (NP (NNP Sydney) (POS 's))
                      (JJ north-western) (NNS suburbs))))))))))))
  (. .)))
Running the parser

Output (2)

nsubj(are-4, Scores-1)
prep_of(Scores-1, properties-3)
root(ROOT-0, are-4)
amod(threat-8, extreme-6)
nn(threat-8, fire-7)
prep_under(are-4, threat-8)
mark(continues-13, as-9)
det(blaze-12, a-10)
amod(blaze-12, huge-11)
nsubj(continues-13, blaze-12)
xsubj(advance-15, blaze-12)
advcl(are-4, continues-13)
aux(advance-15, to-14)
xcomp(continues-13, advance-15)
poss(suburbs-20, Sydney-17)
amod(suburbs-20, north-western-19)
prep_through(advance-15, suburbs-20)
Other languages

./lexparser-lang.sh
Usage: lexparser-lang.sh lang len grammar out_file FILE...

lang : Language to parse (Arabic, English, Chinese, German, French)
len : Maximum length of the sentences to parse
grammar : Serialized grammar file
        (see grammar/ directory)
out_file : Prefix for the output filename
FILE : List of files to parse

To set additional parser options, modify parse_opts
in lexparser_lang.def

Parser memory limit is currently: 10g
Different sources:

- README.txt
- dependencies_manual.pdf
- javadoc/ directory
Options

There are a variety of options dealing with internal details

Some options you might find useful:

- `-sentences newline ...` give one sentence per line (i.e., use your own segmentation)
- `-tokenized` ... assumes white-space separated tokens (so combine MWEs with, e.g., underscores)
  - probably also add: `-escaper` `edu.stanford.nlp.process.PTBEscapingProcessor`
    (to handle PTB token conventions)
- For the PCFG parser: `-printPCFGkBest n ...` prints $n$-best parses
Options

FAQ

See the FAQ for:

- forcing your own POS tags
- memory-handling (for long sentences)
- convert other parser output to dependencies
- using the caseless parser for, e.g., tweets
Use the -outputFormatOptions option with these:

- basicDependencies
- collapsedDependencies
- CCPropagatedDependencies
- treeDependencies
- nonCollapsedDependencies
- nonCollapsedDependenciesSeparated
Options

Dependencies (ex.)

```
java -mx200md -cp "$scriptdir/*:" \
edu.stanford.nlp.parser.lexparser.LexicalizedParser \
-outputFormat "typedDependencies" \
-outputFormatOptions "nonCollapsedDependencies" \
edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz \
data/testsent.txt
```

Loading parser from serialized file
edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz
... done [1.4 sec].
Parsing file: data/testsent.txt
Options

Dependencies (ex., cont.)

nsubj(are-4, Scores-1)
prep(Scores-1, of-2)
pobj(of-2, properties-3)
root(ROOT-0, are-4)
prep(are-4, under-5)
amod(threat-8, extreme-6)
nn(threat-8, fire-7)
pobj(under-5, threat-8)
mark(continues-13, as-9)
det(blaze-12, a-10)
amod(blaze-12, huge-11)
nsubj(continues-13, blaze-12)
xsubj(advance-15, blaze-12)
advcl(are-4, continues-13)
adv(advance-15, to-14)
xcomp(continues-13, advance-15)
prep(advance-15, through-16)
poss(suburbs-20, Sydney-17)
possessive(Sydney-17, 's-18)
amod(suburbs-20, north-western-19)
pobj(through-16, suburbs-20)
Parser training

We will not cover this in detail

An example from the FAQ:

java -mx1g \
edu.stanford.nlp.parser.lexparser.LexicalizedParser \-PCFG -vMarkov 1 -uwm 0 \-headFinder edu.stanford.nlp.trees.LeftHeadFinder \-train train.txt -test test.txt > output.txt

See also bin/makeSerialized.csh