

# Corpus Linguistics (L615)

## Automatic POS and Syntactic Annotation

Markus Dickinson

Department of Linguistics, Indiana University  
Spring 2009

Corpus Linguistics  
Automatic POS and  
Syntactic  
Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood  
Estimation  
Available POS Taggers  
Parsers  
Basics  
Available parsers

## Where we're going

We're going to look at POS tagging & parsing

- ▶ A little bit about how they work
- ▶ More emphasis on getting a few of them working
  - ▶ We'll focus on English, but some of the principles are the same for (some) other languages

Some taggers/parsers have *pre-built* models and other can be *trained*

- ▶ To train them, you need annotated data
- ▶ Of course, you'll want to consider what the annotation scheme is when using a tagger or parser

Corpus Linguistics  
Automatic POS and  
Syntactic  
Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood  
Estimation  
Available POS Taggers  
Parsers  
Basics  
Available parsers

◀ ▶ ⏪ ⏩ 🔍

1 / 37

◀ ▶ ⏪ ⏩ 🔍

2 / 37

## 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://jones.ling.indiana.edu/wiki/TitleIndex>
  - ▶ Always feel free to add your own experiences to help the next person who wants to use that tool
- ▶ ACL wiki & resources
  - ▶ [http://www.aclweb.org/aclwiki/index.php?title=Main\\_Page](http://www.aclweb.org/aclwiki/index.php?title=Main_Page)
  - ▶ [http://www.aclweb.org/aclwiki/index.php?title=ACL\\_Data\\_and\\_Code\\_Repo](http://www.aclweb.org/aclwiki/index.php?title=ACL_Data_and_Code_Repo)
  - ▶ [http://www.aclweb.org/aclwiki/index.php?title=List\\_of\\_resources\\_by\\_language](http://www.aclweb.org/aclwiki/index.php?title=List_of_resources_by_language)
  - ▶ ACL software registry: <http://registry.dfki.de/>
- ▶ UT wiki: <http://comp.ling.utexas.edu/wiki/>

Corpus Linguistics  
Automatic POS and  
Syntactic  
Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood  
Estimation  
Available POS Taggers  
Parsers  
Basics  
Available parsers

## 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 ...

Corpus Linguistics  
Automatic POS and  
Syntactic  
Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood  
Estimation  
Available POS Taggers  
Parsers  
Basics  
Available parsers

◀ ▶ ⏪ ⏩ 🔍

3 / 37

◀ ▶ ⏪ ⏩ 🔍

4 / 37

## Bigram Tagging

Basic assumption: POS tag only depends on word itself and on the POS tag of the previous word

- ▶ Use lexicon to retrieve **ambiguity class** for words
  - ▶ e.g., word: beginning, ambiguity class: [JJ, NN, VBG]
  - ▶ For unknown words: use heuristics, e.g. all open class POS tags
- ▶ Disambiguation: look for most likely path through possibilities

Corpus Linguistics  
Automatic POS and  
Syntactic  
Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood  
Estimation  
Available POS Taggers  
Parsers  
Basics  
Available parsers

## Bigram Tagging – Counter-Examples

- ▶ start before
  - ▶ start before the course or start before he is done
- ▶ real quick
  - ▶ re-cap real quick or a real quick lunch
- ▶ barely changed
  - ▶ he was barely changed or he barely changed his contents
- ▶ that beginning
  - ▶ that beginning part or that beginning frightened the students or with that beginning early, he was forced ...

Corpus Linguistics  
Automatic POS and  
Syntactic  
Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood  
Estimation  
Available POS Taggers  
Parsers  
Basics  
Available parsers

◀ ▶ ⏪ ⏩ 🔍

5 / 37

◀ ▶ ⏪ ⏩ 🔍

6 / 37

# Maximum Likelihood Estimation

simplest way to calculate such probabilities from a corpus:

$$P_{MLE}(t_n | t_{n-1}) = \frac{C(t_{n-1} t_n)}{C(t_{n-1})}$$

$$P_{MLE}(w_n | t_n) = \frac{C(w_n t_n)}{C(t_n)}$$

- ▶ uses relative frequency
- ▶ maximizes the probabilities of the corpus

- Corpus Linguistics
- Automatic POS and Syntactic Annotation
- POS tagging
  - Bigram tagging
  - Maximum Likelihood Estimation
- Available POS Taggers
- Parsers
  - Basics
  - Available parsers

# Maximum Likelihood Estimation (2)

MLE is not a very good estimator on its own:

- ▶ zero probabilities for unseen events: makes them **impossible**
- ▶ need smoothing or discounting method to give minimal probabilities to unseen events
  - ▶ simplest possibility: learn from *hapax legomena* (words that appear only once)

- Corpus Linguistics
- Automatic POS and Syntactic Annotation
- POS tagging
  - Bigram tagging
  - Maximum Likelihood Estimation
- Available POS Taggers
- Parsers
  - Basics
  - Available parsers

# POS taggers

- ▶ TnT: <http://www.coli.uni-saarland.de/~thorsten/tnt/>
  - ▶ Trainable; models for German & English; installed on jones
- ▶ Decision Tree Tagger: <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
- ▶ LingPipe: <http://alias-i.com/lingpipe/index.html>
  - ▶ Has a variety of NLP modules
- ▶ OpenNLP: <http://opennlp.sourceforge.net/>
  - ▶ Models for English, German, Spanish, & Thai; Has a variety of NLP modules

- Corpus Linguistics
- Automatic POS and Syntactic Annotation
- POS tagging
  - Bigram tagging
  - Maximum Likelihood Estimation
- Available POS Taggers
- Parsers
  - Basics
  - Available parsers

# POS taggers (2)

- ▶ ACOPOST: <http://acopost.sourceforge.net/>
  - ▶ Trainable; integrates different technologies
- ▶ Stanford tagger: <http://nlp.stanford.edu/software/tagger.shtml>
  - ▶ Trainable; models for English, Arabic, Chinese, & German
- ▶ CRFTagger: <http://crftagger.sourceforge.net/>
  - ▶ English
- ▶ Can also use SVMTool (<http://www.isi.upc.edu/~nlp/SVMTool/>) or CRF++ (<http://crfpp.sourceforge.net/>) for tagging sequential data, or fnTBL for classification tasks (<http://www.cs.jhu.edu/~rflorian/fntbl/index.html>)

- Corpus Linguistics
- Automatic POS and Syntactic Annotation
- POS tagging
  - Bigram tagging
  - Maximum Likelihood Estimation
- Available POS Taggers
- Parsers
  - Basics
  - Available parsers

# Example: TnT

Obtaining

We have TnT on jones, so you don't really have to do anything to obtain it.

- ▶ It should already be in your PATH, meaning you can access it anywhere
- ▶ If not, I'll add that for you ...

- Corpus Linguistics
- Automatic POS and Syntactic Annotation
- POS tagging
  - Bigram tagging
  - Maximum Likelihood Estimation
- Available POS Taggers
- Parsers
  - Basics
  - Available parsers

# TnT basics

TnT expects input data in the form one word per line, e.g.,

```
this
is
a
test
sentence
.
```

TnT has 3 main commands that we'll use:

- ▶ Tagging: `tnt`
- ▶ Evaluating: `tnt-diff`
- ▶ Training: `tnt-para`

- Corpus Linguistics
- Automatic POS and Syntactic Annotation
- POS tagging
  - Bigram tagging
  - Maximum Likelihood Estimation
- Available POS Taggers
- Parsers
  - Basics
  - Available parsers

## Example: TnT Tagging

The general form for using TnT is:

```
▶ tnt [model] [file-to-be-tagged] > [outputfile]
```

TnT has a few pre-built models you can use: wsj, susanne, susanne2, & negra (for German)

```
▶ tnt wsj test.tt > test.tts
```

Corpus Linguistics  
Automatic POS and Syntactic Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood Estimation

Available POS Taggers

Parsers  
Basics  
Available parsers

14 / 37

## Example session

```
jones:~/... mdickinson$ tnt wsj test.tt > test.tts
TnT: Trigrams'n'Tags - Statistical Trigram Tagging - Version 2.2c
(C) 1993 - 2001 Thorsten Brants, thorsten@brants.net
Reading n-grams /Users/mdickinson/tnt/models/wsj.tnt
..... (46 uni-, 1417 bi-, 17005 trigrams)
Reading lexicon /Users/mdickinson/tnt/models/wsj.tnt
..... (51460 entries)
Building suffix trie .....
(79400 lowercase, 50567 uppercase)
Estimating lambdas ..... (done)
lambda1 = 1.280337e-01 lambda2 = 2.760844e-01 lambda3 = 5.958819e-01
lam_bi1 = 1.770754e-01 lam_bi2 = 8.229246e-01
suffix theta = 9.744236e-02
Setup: real 00:00:00.41, sys 00:00:00.06, proc 00:00:00.32
Tagging (6 tokens)
0 (0.00%) unknown tokens
avg. 2.83 tags/token
Tagging: real 00:00:00.00, sys 00:00:00.00, proc 00:00:00.00 (inf tok/sec)
```

Corpus Linguistics  
Automatic POS and Syntactic Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood Estimation

Available POS Taggers

Parsers  
Basics  
Available parsers

16 / 37

## Example output file

```
%% TnT statistically tagged file, Sat Mar 14 19:26:05 2009
%% lexicon : /Users/mdickinson/tnt/models/wsj.tnt
%% ngrams : /Users/mdickinson/tnt/models/wsj.tnt
%% corpus : test.tt
%% model : trigrams
%% sparse data : linear interpolation
%% lambda1 = 1.280337e-01 lambda2 = 2.760844e-01 lambda3 = 5.958819e-01
%% unknown mode: statistics of singletons
%% case of characters is significant
%% using suffix trie up to length 10
%% suffix backoff with theta = 9.744236e-02
%% Thorsten Brants, thorsten@brants.net
this DT
is VBZ
a DT
test NN
sentence NN
.
%% 0 (0.00%) unknown tokens
%% avg. 2.83 tags/token
```

Corpus Linguistics  
Automatic POS and Syntactic Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood Estimation

Available POS Taggers

Parsers  
Basics  
Available parsers

18 / 37

## Example: TnT Evaluating

If you have a gold standard file in the same format, you can use `tnt-diff` to compare them

```
▶ tnt-diff [tagged-file] [gold-file]
```

```
jones:~/... mdickinson$ tnt-diff test.tts test.gold
TnT-Diff: Show differences between tagged files - Version 2.2c
(C) 1993 - 2001 Thorsten Brants, thorsten@coli.uni-sb.de
Comparing test.tts and test.gold (6 tokens)
Overall result:
Equal : 5 / 6 ( 83.33%)
Different: 1 / 6 ( 16.67%)
```

Corpus Linguistics  
Automatic POS and Syntactic Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood Estimation

Available POS Taggers

Parsers  
Basics  
Available parsers

20 / 37

## Example: TnT Training

TnT can be trained on any corpus which has POS annotation, giving you more flexibility

- ▶ `tnt-para [corpus-file]`
- ▶ This creates a `.lex` (lexicon) and `.123` (trigram) parameter file

```
jones:~/... mdickinson$ tnt-para test.gold
TnT-Para: Generate trigram parameters from corpus - Version 2.2c
(C) 1993 - 2001 Thorsten Brants, thorsten@coli.uni-sb.de
Reading corpus test.gold (6 tokens)
parameter generation from 6 tokens in 1 files
Writing lexicon (6 tokens)
Lexicon written to file 'test.lex'
Writing n-grams (6 uni-, 6 bi-, 5 trigrams)
n-grams written to file 'test.123'
```

Corpus Linguistics  
Automatic POS and Syntactic Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood Estimation

Available POS Taggers

Parsers  
Basics  
Available parsers

22 / 37

## Example 2: Decision Tree Tagger Obtaining

Let's walk through obtaining, tagging, and training the Decision Tree Tagger

1. Go to the website and download the appropriate tagger.
2. Create a directory and put the tar file within that directory.
3. Likewise, download the tagging scripts file into the same directory.
4. Download the installation script into the same directory.
5. Download whichever parameter files you want into the `lib/` subdirectory.

Then look at the README file.

Corpus Linguistics  
Automatic POS and Syntactic Annotation

POS tagging  
Bigram tagging  
Maximum Likelihood Estimation

Available POS Taggers

Parsers  
Basics  
Available parsers

23 / 37

## Example: Decision Tree Tagger Tagging

The general form:

```
tree-tagger {-options-} <parameter file>
  {<input file> {<output file>}}
```

An example for tagging the file wsj10000.txt

```
bin/tree-tagger -token lib/english.par wsj10000.txt
```

- ▶ Try downloading a text, tokenizing it, and tagging it

## Example: Decision Tree Tagger Training

We won't go through an entire example of training, but this is where your scripting skills pay off.

The general form:

```
train-tree-tagger {options} <lexicon> <open class file>
  <input file> <output file>
```

You need the following files:

- ▶ lexicon: word + tags and [optionally] lemmas
- ▶ open class file: listing of open class tags
- ▶ input file: tagged training data (same as TnT format)
- ▶ output file: stores parameters

## The basics of parsing

- ▶ Function of a parser:
  - ▶ grammar + string → analysis trees
- ▶ Main criteria for evaluating parsers:
  - ▶ Correctness: for every grammar and for every string, every analysis returned by parser is an actual analysis
    - ▶ Correctness w.r.t. our target language is thus dependent upon the grammar we give the parser
  - ▶ Completeness: for every grammar and for every string, every correct analysis is found by the parser
    - ▶ May not always be practical, and we may want only one analysis
  - ▶ Efficiency: storing partial parses is essential in being efficient (to be explained)

## Processing for parsing

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

I'm going to give you just a flavor of this today, to see some of the complexity involved

## Direction of processing I: Top-down

**Goal-driven** processing is Top-down:

- ▶ Start with the start symbol
- ▶ Derive sentential forms.
- ▶ If the string is among the sentences derived this way, it is part of the language.

Problem: Left-recursive rules (e.g.,  $NP \rightarrow NP PP$ ) can give rise to infinite hypotheses

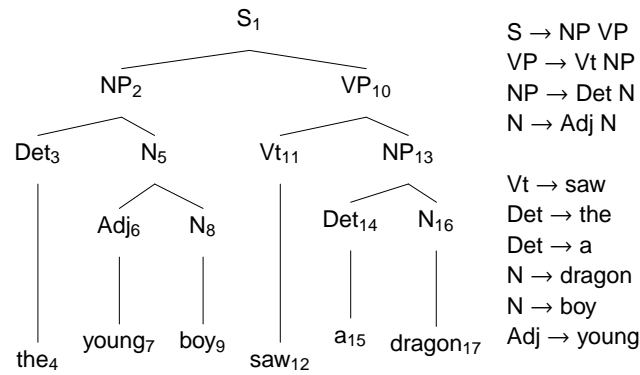
## Direction of processing II: Bottom-up

**Data-driven** processing is Bottom-up:

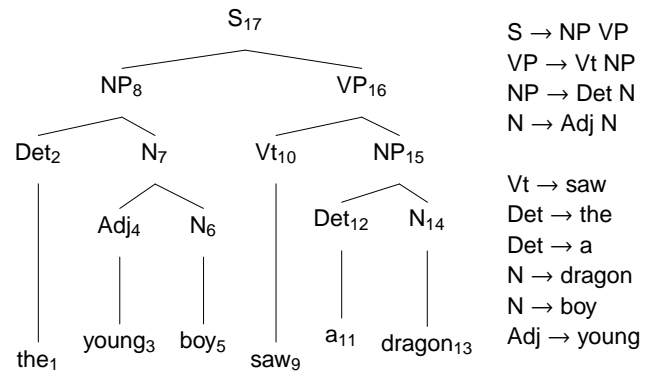
- ▶ Start with the sentence.
- ▶ For each substring, find a grammar rule which covers it.
- ▶ If you finish with a sentence, it is grammatical.

Problem: Epsilon rules ( $N \rightarrow \epsilon$ ) allow us to hypothesize empty categories anywhere in the sentence.

# Top-Down, left-right, depth-first tree traversal



# Bottom-up, left-right, depth-first tree traversal



# 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:**  
<http://nlp.stanford.edu/downloads/lex-parser.shtml>
  - ▶ 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

Dependency parsing is the task of assigning these dependency (grammatical) relations to a sentence

- ▶ Can be done on top of constituency parsing
- ▶ Or can be done on its own

In either case, if we have dependency information, then we know how the elements in the sentence are related to one another ...

# Dependency parsing

- ▶ **MaltParser:**  
<http://w3.msi.vxu.se/~nivre/research/MaltParser.html>
  - ▶ Trainable; models for Swedish, English, & Chinese
- ▶ **MSTParser:** <http://sourceforge.net/projects/mstparser>
  - ▶ Trainable; has some models for English & Portuguese
- ▶ **Link Grammar parser:**  
<http://www.abisource.com/projects/link-grammar/>
  - ▶ English only
- ▶ **CCG parsers:**  
<http://groups.inf.ed.ac.uk/ccg/software.html>
  - ▶ Primarily for English, although can be trained on German CCGbank