LING/C SC 581:
Advanced Computational Linguistics

Lecture 15

Today's Topic

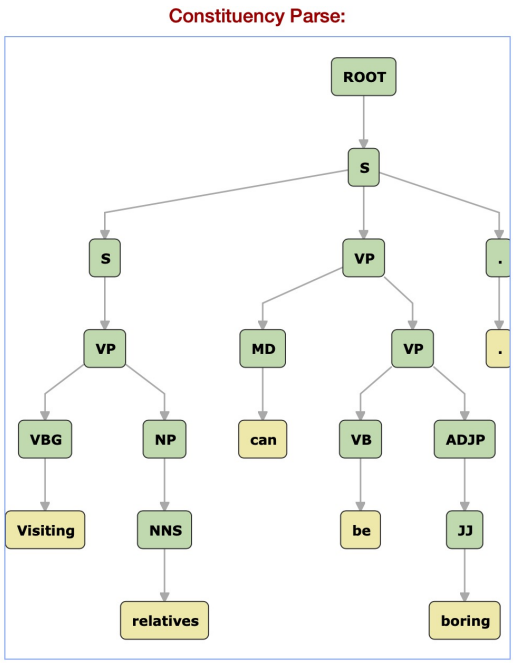
- Last time: introduced CoreNLP
- Today:
  - CoreNLP: coreference
  - install the **Standalone** Stanford Parser (part of CoreNLP).
- Homework 7 (*Easy*)
  - due next Monday night (oops! *Spring Recess*) so Monday 14th ?

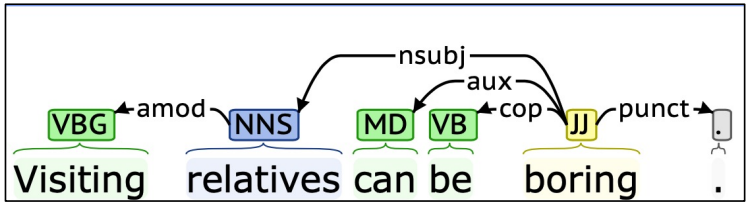
CoreNLP Online

- URL:
  - https://corenlp.run

CoreNLP Online

Visiting relatives can be boring.





We obtained different parses for the sentence, but from two different parsers within CoreNLP …

CoreNLP: command line

- https://stanfordnlp.github.io/CoreNLP/pipeline.html
- Command line example:
  - `java -cp "*" -Xmx5g edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tokenize,ssplit,pos,parser -outputFormat json -file input.txt`
  - `java -cp "*" -Xmx5g edu.stanford.nlp.pipeline.StanfordCoreNLP -props my.props -file input2.txt`

my.props

```
1annotators = tokenize,ssplit,pos,parse
2outputFormat = text
3output.prettyPrint = True
4output.constituencyTree = oneline
```

CoreNLP: command line, –annotators

- https://stanfordnlp.github.io/CoreNLP/annotators.html

Name	Annotator class name	Generated Annotation
tokenize	TokenizerAnnotator	TokensAnnotation (list of tokens); CharacterOffsetBeginAnnotation, CharacterOffsetEndAnnotation, TextAnnotation (for each token)
cleanxml	CleanXmlAnnotator	XmlContextAnnotation
docdate	DocDateAnnotator	DocDateAnnotation
ssplit	WordsToSentencesAnnotator	SentencesAnnotation
pos	POSTaggerAnnotator	PartOfSpeechAnnotation
lemma	MorphaAnnotator	LemmaAnnotation
parse	ParserAnnotator	TreeAnnotation, BasicDependenciesAnnotation, CollapsedDependenciesAnnotation, CollapsedCCProcessedDependenciesAnnotation
depparse	DependencyParseAnnotator	BasicDependenciesAnnotation, CollapsedDependenciesAnnotation, CollapsedCCProcessedDependenciesAnnotation
coref	CorefAnnotator	CorefChainAnnotation
dcoref	DeterministicCorefAnnotator	CorefChainAnnotation

CoreNLP: dependencies, –annotators

Property name	Annotator class name	Requirements
tokenize	TokenizerAnnotator	None
cleanxml	CleanXmlAnnotator	`tokenize`
ssplit	WordsToSentenceAnnotator	`tokenize`
docdate	DocDateAnnotator	None
pos	POSTaggerAnnotator	`tokenize, ssplit`
lemma	MorphaAnnotator	`tokenize, ssplit, pos`
ner	NERClassifierCombiner	`tokenize, ssplit, pos, lemma`
regexner	RegexNERAnnotator	`tokenize, ssplit, pos`
sentiment	SentimentAnnotator	`tokenize, ssplit, pos, parse`
parse	ParserAnnotator	`tokenize, ssplit, parse`

depparse	DependencyParseAnnotator	`tokenize, ssplit, pos`
dcoref	DeterministicCorefAnnotator	`tokenize, ssplit, pos, lemma, ner, parse`
coref	CorefAnnotator	`tokenize, ssplit, pos, lemma, ner, parse` (Can also use `depparse`)
relation	RelationExtractorAnnotator	`tokenize, ssplit, pos, lemma, ner, depparse`
natlog	NaturalLogicAnnotator	`tokenize, ssplit, pos, lemma, depparse` (Can also use `parse`)
entitylink	WikiDictAnnotator	`tokenize, ssplit, ner`
kbp	KBPAnnotator	`tokenize, ssplit, pos, lemma, parse, ner, coref` (Can also use `depparse`; `coref` optional)
quote	QuoteAnnotator	`tokenize, ssplit, pos, lemma, ner, depparse, coref`

CoreNLP: command line, –outputFormat

- https://stanfordnlp.github.io/CoreNLP/cmdline.html#output
- "**text**": An ad hoc human-readable text format. Tokens, s-expression parse trees, relation(head, dep) dependencies. Output file extension is .out. This is the default output format only if the XMLOutputter is unavailable.
- "**xml**": An XML format with accompanying XSLT stylesheet, which allows web browser rendering. Output file extension is .xml. This is the default output format, unless the XMLOutputter is unavailable.
- "**json**": JSON. Output file extension is .json. 'Nuf said.
- "**conll**": A tab-separated values (TSV) format. Output extension is .conll. This output format usually only gives a partial view of an Annotation and doesn't correspond to any particular CoNLL format. By default, the columns written are: idx, word, lemma, pos, ner, headidx, deprel. You can customize which fields are written with the output.columns property. Its value is a comma-separated list of output key names, where the names are ones understood by AnnotationLookup.KeyLookup. Available names include the seven used in the default output and others such as shape, speaker. For instance, you can write out just tokenized text, with one token per line and a blank line between sentences by using -output.columns word. Alternatively, if you give the property output.prettyPrint = false to this outputter, it will print one sentence per line output with the selected fields separated by slash (/) characters. You can hence use this option to write tokenized text, one sentence per line with the options -outputFormat conll -output.columns word -output.prettyPrint false.
- "**conllu**": CoNLL-U output format, another tab-separated values (TSV) format, with particular extended features. Output extension is .conllu. This representation may give only a partial view of an Annotation.

Other output options:

- **output.prettyPrint** : Boolean. Whether to pretty print certain annotations (more friendly to humans; less space efficient.
- **output.constituencyTree** : String. Style of constituency tree printing to be used. One known to TreePrint.
- **output.dependencyTree** : String. Style of dependency tree printing to be used. One known to TreePrint.
- https://nlp.stanford.edu/nlp/javadoc/javanlp-3.5.0/edu/stanford/nlp/trees/TreePrint.html
  - Known formats are: oneline, **penn**, latexTree, xmlTree, words, wordsAndTags, rootSymbolOnly, dependencies, typedDependencies, typedDependenciesCollapsed, collocations, semanticGraph, conllStyleDependencies, conll2007.

Coreference Resolution

System	Language	Preprocessing Time	Coref Time	Total Time	F1 Score
Deterministic	English	3.87s	0.11s	3.98s	49.5
Statistical	English	0.48s	1.23s	1.71s	56.2
Neural	English	3.22s	4.96s	8.18s	60.0
Deterministic	Chinese	0.39s	0.16s	0.55s	47.5
Neural	Chinese	0.42s	7.02s	7.44s	53.9

Coreference Resolution: dcoref

- This is a multi-pass sieve rule-based coreference system. See the Stanford Deterministic Coreference Resolution System page for usage and more details.
- Example:
  - In the directory `stanford-corenlp-4.4.0`
  - `java -cp '*' -Xmx5g edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tokenize,ssplit,pos,lemma,ner,parse,dcoref -file input3.txt`

```
1 John saw pictures of himself with Pete.¶
2 John saw pictures of him.¶
3 John believes that Mary likes him.¶
4 John believes himself to like Mary.¶
5 John believes him to like Mary too.¶
6 John is sure that he likes Mary.¶
7 John's picture of him is nice.¶
8 The men think that pictures of each other will be available to Mary.¶
```

Coreference Resolution: dcoref

1 John saw pictures of himself with Pete.¶
2 John saw pictures of him.¶
3 John believes that Mary likes him.¶
4 John believes himself to like Mary.¶
5 John believes him to like Mary too.¶
6 John is sure that he likes Mary.¶
7 John's picture of him is nice.¶
8 The men think that pictures of each other will be available to Mary.¶

Coreference set:
(1,5,[5,6]) -> (1,1,[1,2]), that is: "himself" -> "John"
(2,1,[1,2]) -> (1,1,[1,2]), that is: "John" -> "John"
(2,5,[5,6]) -> (1,1,[1,2]), that is: "him" -> "John"
(3,1,[1,2]) -> (1,1,[1,2]), that is: "John" -> "John"
(3,6,[6,7]) -> (1,1,[1,2]), that is: "him" -> "John"
(4,1,[1,2]) -> (1,1,[1,2]), that is: "John" -> "John"
(4,3,[3,4]) -> (1,1,[1,2]), that is: "himself" -> "John"
(5,1,[1,2]) -> (1,1,[1,2]), that is: "John" -> "John"
(5,3,[3,4]) -> (1,1,[1,2]), that is: "him" -> "John"
(6,1,[1,2]) -> (1,1,[1,2]), that is: "John" -> "John"
(6,5,[5,6]) -> (1,1,[1,2]), that is: "he" -> "John"
(7,1,[1,3]) -> (1,1,[1,2]), that is: "John 's" -> "John"
(7,5,[5,6]) -> (1,1,[1,2]), that is: "him" -> "John"

- **Output:**

Extracted the following NER entity mentions:

John	PERSON	PERSON:0.9968535235647067
Pete	PERSON	PERSON:0.9959526451634645
John	PERSON	PERSON:0.9929680223897125
him	PERSON	-
Mary	PERSON	PERSON:0.995973577656132

Coreference set:
(2,3,[3,6]) -> (1,3,[3,6]), that is: "pictures of him" -> "pictures of himself"

Coreference set:
(4,6,[6,7]) -> (3,4,[4,5]), that is: "Mary" -> "Mary"
(5,6,[6,7]) -> (3,4,[4,5]), that is: "Mary" -> "Mary"
(6,7,[7,8]) -> (3,4,[4,5]), that is: "Mary" -> "Mary"
(8,13,[13,14]) -> (3,4,[4,5]), that is: "Mary" -> "Mary"

Coreference Resolution: coref

- **Neural:**
  - Most accurate but slow neural-network-based coreference resolution for English and Chinese.
- Example:
  - `java -Xmx5g -cp stanford-corenlp-4.0.0.jar:stanford-corenlp-4.0.0-models.jar:* edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tokenize,ssplit,pos,lemma,ner,parse,coref -coref.algorithm neural -file input3.txt`

Coreference Resolution: coref

1 John saw pictures of himself with Pete.¶
2 John saw pictures of him.¶
3 John believes that Mary likes him.¶
4 John believes himself to like Mary.¶
5 John believes him to like Mary too.¶
6 John is sure that he likes Mary.¶
7 John's picture of him is nice.¶
8 The men think that pictures of each other will be available to Mary.¶

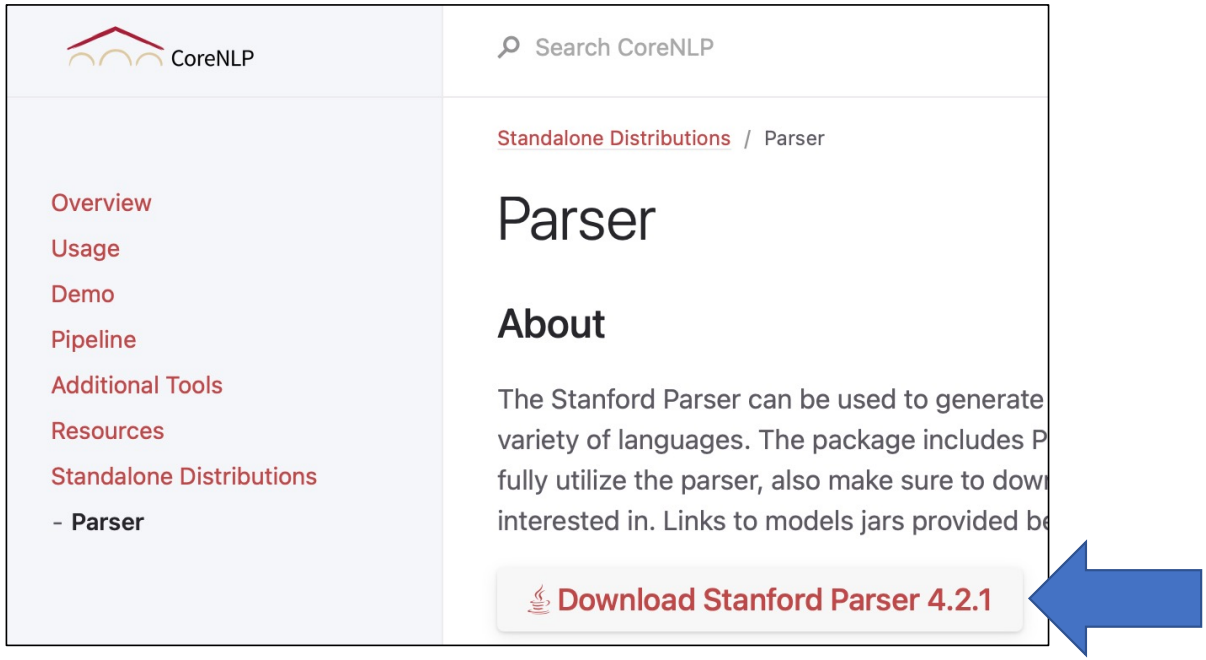
Coreference set:
(4,6,[6,7]) -> (3,4,[4,5]), that is: "Mary" -> "Mary"
(5,6,[6,7]) -> (3,4,[4,5]), that is: "Mary" -> "Mary"
(6,7,[7,8]) -> (3,4,[4,5]), that is: "Mary" -> "Mary"
(8,13,[13,14]) -> (3,4,[4,5]), that is: "Mary" -> "Mary"

- **Output**:

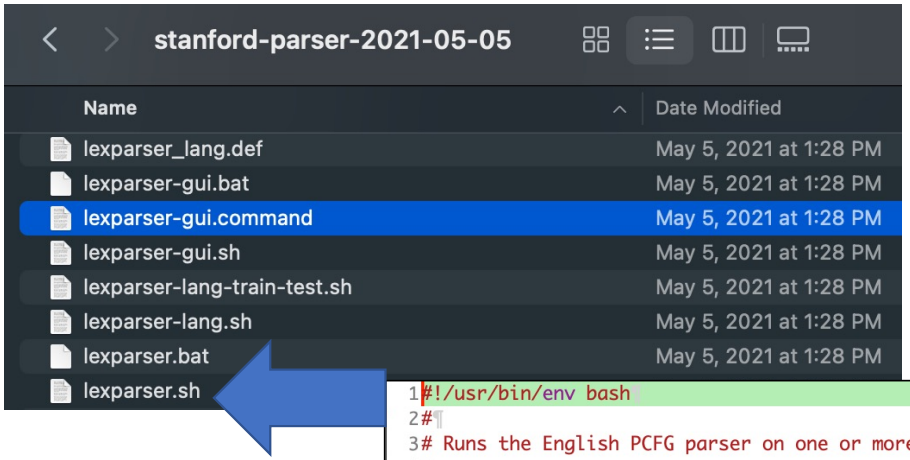
Coreference set:
(1,1,[1,2]) -> (7,1,[1,3]), that is: "John" -> "John 's"
(1,5,[5,6]) -> (7,1,[1,3]), that is: "himself" -> "John 's"
(2,1,[1,2]) -> (7,1,[1,3]), that is: "John" -> "John 's"
(2,5,[5,6]) -> (7,1,[1,3]), that is: "him" -> "John 's"
(3,1,[1,2]) -> (7,1,[1,3]), that is: "John" -> "John 's"
(3,6,[6,7]) -> (7,1,[1,3]), that is: "him" -> "John 's"
(4,1,[1,2]) -> (7,1,[1,3]), that is: "John" -> "John 's"
(4,3,[3,4]) -> (7,1,[1,3]), that is: "himself" -> "John 's"
(5,1,[1,2]) -> (7,1,[1,3]), that is: "John" -> "John 's"
(5,3,[3,4]) -> (7,1,[1,3]), that is: "him" -> "John 's"
(6,1,[1,2]) -> (7,1,[1,3]), that is: "John" -> "John 's"
(6,5,[5,6]) -> (7,1,[1,3]), that is: "he" -> "John 's"
(7,5,[5,6]) -> (7,1,[1,3]), that is: "him" -> "John 's"

CoreNLP: Standalone Parser

- https://stanfordnlp.github.io/CoreNLP/parser-standalone.html



CoreNLP: Standalone Parser



In directory: `stanford-parser-2021-05-05`
Command line:
`./lexparser.sh data/mydata.txt`

`lexparser.sh`

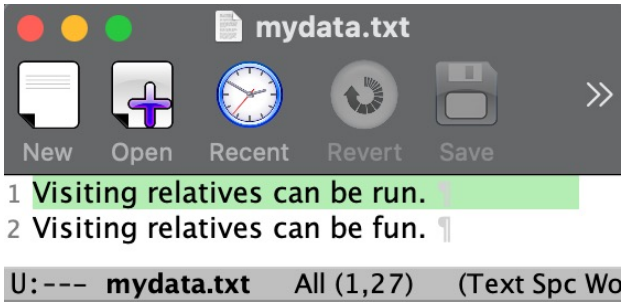
```
#!/usr/bin/env bash
#
# Runs the English PCFG parser on one or more files, printing trees only

if [ ! $# -ge 1 ]; then
  echo Usage: `basename $0` 'file(s)'
  echo
  exit
fi

scriptdir=`dirname $0`

java -mx200m -cp "$scriptdir/*:" edu.stanford.nlp.parser.lexparser.LexicalizedParser \
 -outputFormat "penn,typedDependencies" edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz $*
```

`data/mydata.txt`



CoreNLP: Standalone Parser

```
[main] INFO edu.stanford.nlp.parser.lexparser.LexicalizedParser - Loading
parser from serialized file
edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz ... done [0.5 sec].

Parsing file: data/mydata.txt

Parsing [sent. 1 len. 6]: Visiting relatives can be run .

(ROOT
  (S
    (S
      (VP (VBG Visiting)
        (NP (NNS relatives))))
    (VP (MD can)
      (VP (VB be)
        (VP (VBN run))))
    (. .)))
csubj:pass(run-5, Visiting-1)
obj(Visiting-1, relatives-2)
aux(run-5, can-3)
aux:pass(run-5, be-4)
root(ROOT-0, run-5)
```

```
Parsing [sent. 2 len. 6]: Visiting relatives can be fun .

(ROOT
  (S
    (S
      (VP (VBG Visiting)
        (NP (NNS relatives))))
    (VP (MD can)
      (VP (VB be)
        (ADJP (JJ fun))))
    (. .)))
csubj(fun-5, Visiting-1)
obj(Visiting-1, relatives-2)
aux(fun-5, can-3)
cop(fun-5, be-4)
root(ROOT-0, fun-5)

Parsed file: data/mydata.txt [2 sentences].
Parsed 12 words in 2 sentences (44.78 wds/sec; 7.46 sents/sec).
```

Question 1

- https://nlp.stanford.edu/software/parser-faq.html
- Modify `lexparser.sh` to give the top two parses.

14. **Can I obtain multiple parse trees for a single input sentence?**

    Yes, for the PCFG parser (only). With a PCFG parser, you can give the option `-printPCFGkBest n` and it will print the *n* highest-scoring parses for a sentence. They can be printed either as phrase structure trees or as typed dependencies in the usual way via the `-outputFormat` option, and each receives a score (log probability). The *k* best parses are extracted efficiently using the algorithm of Huang and Chiang (2005).

Question 1

- Run it on the sentence:
  - Visiting relatives can be fun.
- Explain the two parses.

Question 2

- What about the following ambiguous sentence?
  - *John saw the man with a telescope*
- Explain the PP attachment ambiguity.
- Does the parser give the correct two parses?

Question 3

- In principle, how many (syntactic) parses should there be for the following sentence?
  - John saw the man with a gun with a telescope
- Explain.
- Then modify `lexparser.sh` to give the required number of parses.
- Did it give the right outputs?
- The parses are ranked in order (–logprob).
- Do you think the order is right?