# NLP Tools

L715/B659

Dept. of Linguistics, Indiana University

Fall 2016

# Tools



**Step 1:** See a bit more of what NLTK has to offer

- cf. the “all in python” approach

**Step 2:** Look at the Stanford NLP tools (noting that there are many others available)

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

Many tools come bundled as part of the CoreNLP package, which we’ll work with:

- http://nlp.stanford.edu/software/corenlp.shtml
- Python wrappers for CoreNLP: http://stanfordnlp.github.io/CoreNLP/other-languages.html

**Step 3:** Work on a practical problem

- Use tools to get output, from which features can be extracted for classification

# NLTK POS tagging



```
>>> import nltk
>>> text = nltk.word_tokenize("And now for something \
    completely different")
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'),
 ('something', 'NN'), ('completely', 'RB'),
 ('different', 'JJ')]
```

(http://www.nltk.org/book/ch05.html)

# NLTK parsing



Parsing in NLTK is a bit trickier, as it generally requires either:

- a context-free grammar (http://stackoverflow.com/questions/6115677/english-grammar-for-parsing-in-nltk); or
- another third-party package

We'll skip to exploring other tools

- Many other tools also have Python wrappers

(http://www.nltk.org/book/ch08.html)

# CoreNLP tools



The tools include:

- Tokenizer (`tokenize`)
- Sentence splitter (`ssplit`)
- Part-of-speech (POS) tagger (`pos`)
- Named entity recognizer (NER) (`ner`)
- Parser (`parse`)
- Coreference resolution system (`dcoref`)
- Sentiment analysis system (`sentiment`)
- Mention detection (`mention`)
- Bootstrapped pattern learning tools

# Out-of-the-box



```
./corenlp.sh
java -mx5g -cp "./*" edu.stanford.nlp.pipeline.StanfordCoreNLP
[main] INFO edu.stanford.nlp.pipeline.StanfordCoreNLP - Searching for reso
[main] INFO edu.stanford.nlp.pipeline.StanfordCoreNLP - Searching for reso
... Adding annotator tokenize
... Adding annotator ssplit
... Adding annotator pos
... Adding annotator lemma
... Adding annotator ner ...
Reading TokensRegex rules  ...
... Adding annotator parse ...
done [0.4 sec].
... Adding annotator mention
... Adding annotator coref
```

# Interactive shell



```
Entering interactive shell. Type q RETURN or EOF to quit.
NLP> Analyze this sentence!
Sentence #1 (4 tokens):
Analyze this sentence!
[Text=Analyze CharacterOffsetBegin=0 CharacterOffsetEnd=7 PartOfSpeech=VB
[Text=this CharacterOffsetBegin=8 CharacterOffsetEnd=12 PartOfSpeech=DT Le
[Text=sentence CharacterOffsetBegin=13 CharacterOffsetEnd=21 PartOfSpeech=
[Text=! CharacterOffsetBegin=21 CharacterOffsetEnd=22 PartOfSpeech=. Lemma
(ROOT
  (S
    (VP (VB Analyze)
      (NP (DT this) (NN sentence)))
    (. !)))

root(ROOT-0, Analyze-1)
det(sentence-3, this-2)
dobj(Analyze-1, sentence-3)
punct(Analyze-1, !-4)
```

# Specifying annotators



1. Pass in arguments to `corenlp.sh`

   ```
   $ ./corenlp.sh -annotators tokenize,ssplit,pos,lemma,ner
     -file input.txt
   ```

2. Java properties file
   - Content of `sampleProps.properties`:

     ```
     annotators = tokenize, ssplit, pos
     outputExtension = .output
     file = input.txt
     ```

   - Command: `./corenlp.sh -props sampleProps.properties -file input.txt`

http://stanfordnlp.github.io/CoreNLP/cmdline.html

# Working from another directory



To call the files, make sure the classpath (`cp`) is set properly, e.g.,

- Command line: `java -cp "/path/to/stanfordcorenlp/*" -Xmx2g edu.stanford.nlp.pipeline.StanfordCoreNLP -file input.txt`
  - Note the asterisk
  - Make sure `input.txt` is present

# Output format

## One token

```
<token id="1">
  <word>Stanford</word>
  <lemma>Stanford</lemma>
  <CharacterOffsetBegin>0</CharacterOffsetBegin>
  <CharacterOffsetEnd>8</CharacterOffsetEnd>
  <POS>NNP</POS>
  <NER>ORGANIZATION</NER>
</token>
```

# Output format

## Constituency parse

```
<parse>(ROOT (S (NP (NNP Stanford) (NNP University))
(VP (VBZ is) (ADJP (JJ located) (PP (IN in) (NP (NNP
 California))))) (. .))) </parse>
```

# Output format

## Dependency parse

```
<basic-dependencies>
  <dep type="nn">
    <governor idx="2">University</governor>
    <dependent idx="1">Stanford</dependent>
  </dep>
  <dep type="nsubj">
    <governor idx="4">located</governor>
    <dependent idx="2">University</dependent>
  </dep>
  ...
</dependencies>
```

# Output format

## Coreference

```
<coreference>
  <coreference>
    <mention representative="true">
      <sentence>1</sentence>
      <start>1</start>
      <end>3</end>
      <head>2</head>
    </mention>
    <mention>
      <sentence>2</sentence>
      <start>1</start>
      <end>2</end>
      <head>1</head>
    </mention>
    ...
  </coreference>
</coreference>
```

(See: http://stanfordnlp.github.io/CoreNLP/faq.html for

# Efficiency note



**CoreNLP runs out of memory?**

*Either give CoreNLP more memory, use fewer annotators, or give CoreNLP smaller documents. Nearly all our annotators load large model files which use lots of memory. Running the full CoreNLP pipeline requires the sum of all these memory requirements. Typically, this means that CoreNLP needs about 2GB to run the entire pipeline. Additionally, the coreference module operates over an entire document. Unless things are size-limited, as either sentence length or document size increases, processing time and space increase without bound.*

# Outputting as text



```
./corenlp.sh -file input.txt -outputFormat text
```

```
[Text=Stanford CharacterOffsetBegin=0 CharacterOffsetEnd=8
PartOfSpeech=NNP Lemma=Stanford
NamedEntityTag=ORGANIZATION] ...

(ROOT
  (S
    (NP (NNP Stanford) (NNP University))
    (VP (VBZ is)
      (ADJP (JJ located)
        (PP (IN in)
          (NP (NNP California)))))
    (. .)))

root(ROOT-0, located-4)
compound(University-2, Stanford-1)
nsubj(located-4, University-2)
cop(located-4, is-3)
...
```

# Different annotator options



Each of the annotators has various options associated with it, e.g.,

- `clean.xmltags`: Discard xml tag tokens that match this regular expression. For example, .* will discard all xml tags
- `quote.singleQuotes`: “true” or “false”, indicating whether or not to consider ’ tokens to be quotation marks (default=false).
- `regexner.mapping`: Comma separated list of mapping files to use. Each mapping file is a tab delimited file ...
- `parse.flags`: flags to use when loading the parser model. ...

# External code



There are some extensions to the CoreNLP package, including many wrappers in various languages

- Java, .NET, Python, Ruby, Perl, Scala, Clojure, JavaScript

http://stanfordnlp.github.io/CoreNLP/extensions.html

# Understanding the categories



For POS definitions:

- http://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html

For constituent syntactic phrases:

- table 3 of: https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html

For dependency categories:

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

# Practice problem

## Step 1: Obtain data



1. Obtain data
   - 1.1 Go to: http://pan.webis.de/clef16/pan16-web/author-profiling.html
   - 1.2 Download: i. corpus & ii. software
   - 1.3 Unpack both, including at least one of the language .zip files in the corpus
   - 1.4 Run the software, e.g.,

```
$ java -jar TwitterDownloader.jar -data \
  /PATH/TO/pan16-author-profiling-training-dataset-english-
```

   - You'll probably want to cut off the program after just a few files (for now) ... as long as you get mostly non-null files

# Practice problem

## Steps 2–5: Work with data



2. Write a script to get the text (e.g., with BeautifulSoup)
3. Provide linguistic annotations for the text from the Stanford tools
4. Begin to extract linguistically-motivated features from the files you're working with
5. If time: Begin to do proper classification . . .

# Old practice problem

1. Download & unpack the ICE-Nigeria corpus: http://sourceforge.net/projects/ice-nigeria/
   - Note: part of the International Corpus of English (ICE): https://sourceforge.net/projects/ice-nigeria/
   - (If there's another corpus you're more enthused about working with, feel free to do so.)
2. Under `ice-nig/txt/written/`, identify a set of files to work with
3. Provide linguistic annotations from the Stanford tools
   - Consider the task of genre classification again: what linguistic annotations would be helpful?
4. Begin to extract linguistically-motivated features from the files you're working with (using your favorite programming language)
5. If time: Begin to do proper classification . . .