# NLTK in 20 minutes

## A sprint thru Python's Natural Language ToolKit

# Why Text Processing?

- sentiment analysis
- spam filtering
- plagariasm detection / document similarity
- document categorization / topic detection
- phrase extraction, summarization
- smarter search
- simple keyword frequency analysis

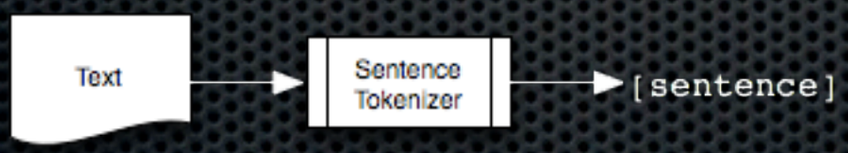
# Some NLTK Features

- sentence & word tokenization
- part-of-speech tagging
- chunking & named entity recognition
- text classification
- many included corpora

# Sentence Tokenization

```
>>> from nltk.tokenize import sent_tokenize
>>> sent_tokenize("Hello SF Python. This is NLTK.")
['Hello SF Python.', 'This is NLTK.']
```

```
>>> sent_tokenize("Hello, Mr. Anderson. We missed you!")
['Hello, Mr. Anderson.', 'We missed you!']
```



# Word Tokenization

```
>>> from nltk.tokenize import word_tokenize
>>> word_tokenize('This is NLTK.')
['This', 'is', 'NLTK', '.']
```

# What's a Word?

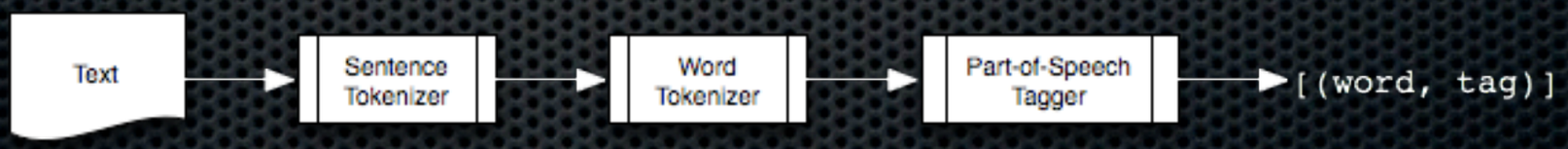
```
>>> word_tokenize("What's up?")
['What', "'s", 'up', '?']
>>> from nltk.tokenize import wordpunct_tokenize
>>> wordpunct_tokenize("What's up?")
['What', "'", 's', 'up', '?']
```

Learn More: http://text-processing.com/demo/tokenize/

# Part-of-Speech Tagging

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

Tags List: http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

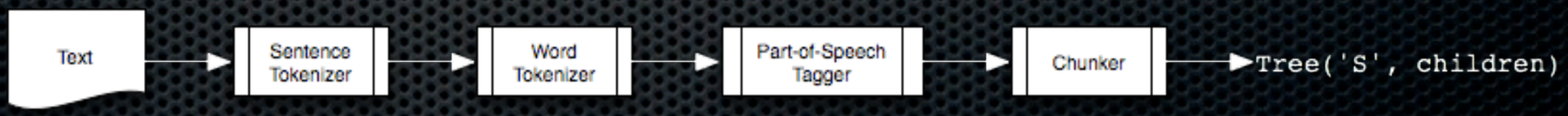


# Why Part-of-Speech Tag?

- word definition lookup (WordNet, WordNik)
- fine-grained text analytics
- part-of-speech specific keyword analysis
- chunking & named entity recognition (NER)

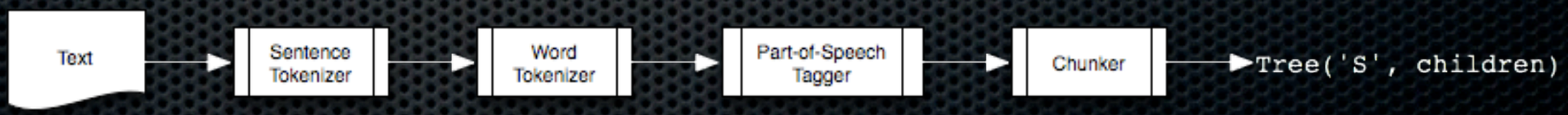
# Chunking & NER

```
>>> from nltk.chunk import ne_chunk
>>> ne_chunk(pos_tag(word_tokenize('My name is Jacob
Perkins.')))
Tree('S', [('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'),
Tree('PERSON', [('Jacob', 'NNP'), ('Perkins', 'NNP')]),
('.', '.')])
```

# NER not perfect

```
>>> ne_chunk(pos_tag(word_tokenize('San Francisco is
foggy.')))
Tree('S', [Tree('GPE', [('San', 'NNP')]), Tree('PERSON',
[('Francisco', 'NNP')]), ('is', 'VBZ'),  ('foggy', 'NN'),
('.', '.')])
```

# Text Classification

```
def bag_of_words(words):
    return dict([(word, True) for word in words])

>>> feats = bag_of_words(word_tokenize("great movie"))
>>> import nltk.data
>>> classifier = nltk.data.load('classifiers/
movie_reviews_NaiveBayes.pickle')
>>> classifier.classify(feats)
'pos'
```

# Classification Algos in NLTK

- Naive Bayes
- Maximum Entropy / Logistic Regression
- Decision Tree
- SVM (coming soon)

# NLTK-Trainer

- https://github.com/japerk/nltk-trainer
- command line scripts
- train custom models
- analyze corpora
- analyze models against corpora

# Train a Sentiment Classifier

```
$ ./train_classifier.py movie_reviews --instances paras
loading movie_reviews
2 labels: ['neg', 'pos']
2000 training feats, 2000 testing feats
training NaiveBayes classifier
accuracy: 0.967000
neg precision: 1.000000
neg recall: 0.934000
neg f-measure: 0.965874
pos precision: 0.938086
pos recall: 1.000000
pos f-measure: 0.968054
dumping NaiveBayesClassifier to ~/nltk_data/classifiers/
movie_reviews_NaiveBayes.pickle
```

# Notable Included Corpora

- movie_reviews: pos & neg categorized IMDb reviews
- treebank: tagged and parsed WSJ text
- treebank_chunk: tagged and chunked WSJ text
- brown: tagged & categorized english text
- 60 other corpora in many languages

# Other NLTK Features

- clustering
- metrics
- parsing
- stemming
- WordNet
- ... and a lot more

# Other Python NLP Libraries

- pattern: http://www.clips.ua.ac.be/pages/pattern
- scikits.learn: http://scikit-learn.sourceforge.net/stable/
- fuzzywuzzy: https://github.com/seatgeek/fuzzywuzzy