# Applied Text Mining in Python

## *Basic Natural Language Processing*

# What is Natural Language?

- Language used for everyday communication by humans
  - English
  - 中文
  - हिन्दी
  - ру́сский язы́к
  - español

c u 2nite

compared to the artificial computer languages

# What is Natural Language Processing?

- Any computation, manipulation of natural language

- Natural languages evolve
  - new words get added
  - old words lose popularity
  - meanings of words change
  - language rules themselves may change

selfie

thou

learn

position of verbs in sentences!

# NLP Tasks: A Broad Spectrum

- Counting words, counting frequency of words
- Finding sentence boundaries
- Part of speech tagging
- Parsing the sentence structure
- Identifying semantic roles
- Identifying entities in a sentence
- Finding which pronoun refers to which entity

and much more …

# Applied Text Mining in Python

*Basic NLP tasks with NLTK*

# An Introduction to NLTK

- NLTK: Natural Language Toolkit
- Open source library in Python
- Has support for most NLP tasks
- Also provides access to numerous text corpora

# Let's set it up first!

```
>>> import nltk
```

- **Let's get some text corpora**

```
>>> nltk.download()
```

```
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

# Let's set it up first! (2)

```
>>> text1
<Text: Moby Dick by Herman Melville 1851>


>>> sents()
sent1: Call me Ishmael .
sent2: The family of Dashwood had long been settled in Sussex .
sent3: In the beginning God created the heaven and the earth .
sent4: Fellow - Citizens of the Senate and of the House of Representatives :
sent5: I have a problem with people PMing me to lol JOIN
sent6: SCENE 1 : [ wind ] [ clop clop clop ] KING ARTHUR : Whoa there !
sent7: Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov.
29 .
sent8: 25 SEXY MALE , seeks attrac older single lady , for discreet encounters .
sent9: THE suburb of Saffron Park lay on the sunset side of London , as red and ragged as
a cloud of sunset .


>>> sent1
['Call', 'me', 'Ishmael', '.']
```

# Simple NLP tasks (1)

- **Counting vocabulary of words**

```
>>> text7
<Text: Wall Street Journal>
>>> sent7
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the',
'board', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']
>>> len(sent7)
18
>>> len(text7)
100676
>>> len(set(text7))
12408
```

# Simple NLP tasks (2)

```
>>> list(set(text7))[:10]
[u'Mortimer', u'foul', u'Heights', u'four', u'spiders', u'railing', u'centimeter',
u'Until', u'payoff', u'Germany-based']
```

- **Frequency of words**

```
>>> dist = FreqDist(text7)
>>> len(dist)
12408
>>> vocab1 = dist.keys()
>>> list(vocab1)[:10]
[u'Mortimer', u'foul', u'Heights', u'four', u'spiders', u'railing', u'centimeter',
u'Until', u'payoff', u'Germany-based']
>>> dist[u'four']
20
>>> freqwords = [w for w in vocab1 if len(w) > 5 and dist[w] > 100]
>>> freqwords
[u'million', u'shares', u'market', u'president', u'trading', u'billion', u'company',
u'program', u'because']
```

# Normalization and Stemming

- **Different forms of the same "word"**

```
>>> input1 = "List listed lists listing listings"
>>> words1 = input1.lower().split(' ')
>>> words1
['list', 'listed', 'lists', 'listing', 'listings']


>>> porter = nltk.PorterStemmer()
>>> [porter.stem(t) for t in words1]
[u'list', u'list', u'list', u'list', u'list']
```

# Lemmatization

```
>>> udhr = nltk.corpus.udhr.words('English-Latin1')
>>> udhr[:20]
['Universal', 'Declaration', 'of', 'Human', 'Rights', 'Preamble', 'Whereas',
'recognition', 'of', 'the', 'inherent', 'dignity', 'and', 'of', 'the', 'equal', 'and',
'inalienable', 'rights', 'of']

>>> [porter.stem(t) for t in udhr[:20]]
[u'Univers', u'Declar', u'of', u'Human', u'Right', u'Preambl', u'Wherea', u'recognit',
u'of', u'the', u'inher', u'digniti', u'and', u'of', u'the', u'equal', u'and',
u'inalien', u'right', u'of']
```

- **Lemmatization: Stemming, but resulting stems are all valid words**

```
>>> WNlemma = nltk.WordNetLemmatizer()
>>> [WNlemma.lemmatize(t) for t in udhr[:20]]
['Universal', 'Declaration', 'of', 'Human', 'Rights', 'Preamble', 'Whereas',
'recognition', 'of', 'the', 'inherent', 'dignity', 'and', 'of', 'the', 'equal', 'and',
'inalienable', u'right', 'of']
```

# Tokenization

- **Recall splitting a sentence into words / tokens**

```
>>> text11 = "Children shouldn't drink a sugary drink before bed."
>>> text11.split(' ')
['Children', "shouldn't", 'drink', 'a', 'sugary', 'drink', 'before',
'bed.’]
```

- **NLTK has an in-built tokenizer**

```
>>> nltk.word_tokenize(text11)
['Children', 'should', "n't", 'drink', 'a', 'sugary', 'drink', 'before',
'bed', '.']
```

# Sentence Splitting

- **How would you split sentences from a long text string?**

```
>>> text12 = "This is the first sentence. A gallon of milk in the U.S. costs
$2.99. Is this the third sentence? Yes, it is!"
```

- **NLTK has an in-built sentence splitter too!**

```
>>> sentences = nltk.sent_tokenize(text12)
>>> len(sentences)
4
>>> sentences
['This is the first sentence.', 'A gallon of milk in the U.S. costs
$2.99.', 'Is this the third sentence?', 'Yes, it is!']
```

# Take Home Concepts

- NLTK is a widely used toolkit for text and natural language processing
- NLTK gives access to many corpora and handy tools
- Sentence splitting, tokenization, and lemmatization are important, and non-trivial, pre-processing tasks

# Applied Text Mining in Python

*Advanced NLP tasks with NLTK*

# NLP Tasks

- Counting words, counting frequency of words
- Finding sentence boundaries
- Part of speech tagging
- Parsing the sentence structure
- Identifying semantic role labeling
- Named Entity Recognition
- Co-reference and pronoun resolution

# Part-of-speech (POS) Tagging

- **Recall high school grammar: nouns, verbs, adjectives, …**
- **Many more tags or word classes than just these**

| Tag | Word class | Tag | Word class | Tag | Word class |
|---|---|---|---|---|---|
| CC | Conjunction | JJ | Adjective | PRP | Pronoun |
| CD | Cardinal | MD | Modal | RB | Adverb |
| DT | Determiner | NN | Noun | SYM | Symbol |
| IN | Preposition | POS | Possessive | VB | Verb |

…

```
>>> import nltk
>>> nltk.help.upenn_tagset('MD')
MD: modal auxiliary
    can cannot could couldn't dare may might must need ought
    shall should shouldn't will would
```

# POS Tagging with NLTK

- **Recall splitting a sentence into words / tokens**

```
>>> text11 = "Children shouldn't drink a sugary drink
before bed."
>>> text13 = nltk.word_tokenize(text11)
```

- **NLTK's Tokenizer**

```
>>> nltk.pos_tag(text13)
[('Children', 'NNP'), ('should', 'MD'), ("n't", 'RB'),
('drink', 'VB'), ('a', 'DT'), ('sugary', 'JJ'), ('drink',
'NN'), ('before', 'IN'), ('bed', 'NN'), ('.', '.')]
```

# Ambiguity in POS Tagging

- **Ambiguity is common in English**

**Visiting aunts can be a nuisance**

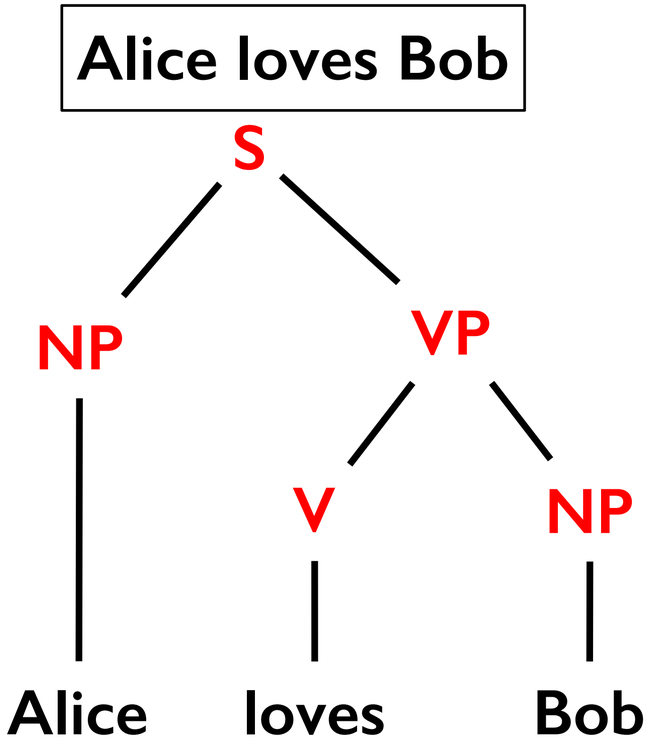
```
>>> text14 = nltk.word_tokenize("Visiting aunts can be a nuisance")
>>> nltk.pos_tag(text14)
[('Visiting', 'VBG'), ('aunts', 'NNS'), ('can', 'MD'), ('be', 'VB'), ('a', 'DT'),
('nuisance', 'NN')]
```

- **Another alternative POS tagging**

```
[('Visiting', 'JJ'), ('aunts', 'NNS'), ('can', 'MD'), ('be', 'VB'), ('a', 'DT'),
('nuisance', 'NN')]
```

# Parsing Sentence Structure

- Making sense of sentences is easy if they follow a well-defined grammatical structure



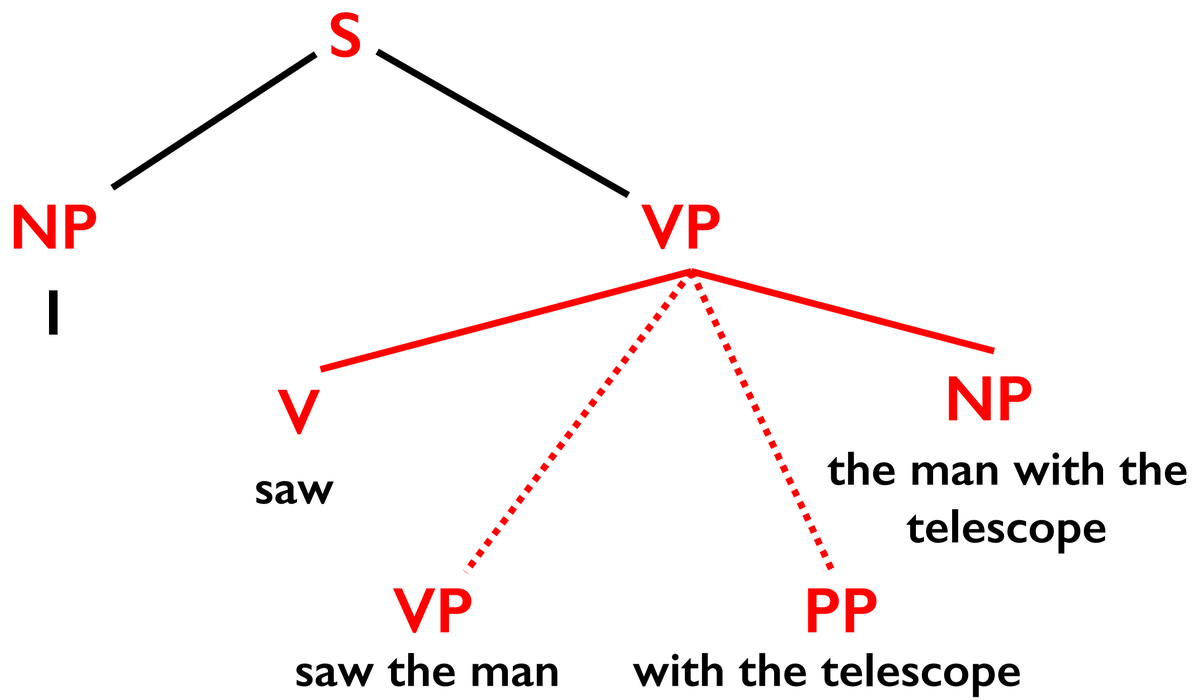

```
>>> text15 = nltk.word_tokenize("Alice loves Bob")
>>> grammar = nltk.CFG.fromstring("""
... S -> NP VP
... VP -> V NP
... NP -> 'Alice' | 'Bob'
... V -> 'loves'
... """)

>>> parser = nltk.ChartParser(grammar)
>>> trees = parser.parse_all(text15)
>>> for tree in trees:
...     print tree
...
(S (NP Alice) (VP (V loves) (NP Bob)))
```

# Ambiguity in Parsing

- Ambiguity may exist even if sentences are grammatically correct!

I saw the man with a telescope

```
S -> NP VP
VP -> V NP | VP PP
PP -> P NP
NP -> DT N | DT N PP | 'I'
DT -> 'a' | 'the'
N -> 'man' | 'telescope'
V -> 'saw'
P -> 'with'
```



# Ambiguity in Parsing

```
>>> text16 = nltk.word_tokenize("I saw the man with a telescope")
>>> grammar1 = nltk.data.load('mygrammar1.cfg')
>>> grammar1
<Grammar with 13 productions>
>>> parser = nltk.ChartParser(grammar1)
>>> trees = parser.parse_all(text16)
>>> for tree in trees:
...   print tree
...
(S
  (NP I)
  (VP
    (VP (V saw) (NP (DT the) (N man)))
    (PP (P with) (NP (DT a) (N telescope)))))
(S
  (NP I)
  (VP
    (V saw)
    (NP (DT the) (N man) (PP (P with) (NP (DT a) (N telescope))))))
```

```
S -> NP VP
VP -> V NP | VP PP
PP -> P NP
NP -> DT N | DT N PP | 'I'
DT -> 'a' | 'the'
N -> 'man' | 'telescope'
V -> 'saw'
P -> 'with'
```

mygrammar1.cfg

# NLTK and Parse Tree Collection

```
>>> from nltk.corpus import treebank
>>> text17 = treebank.parsed_sents('wsj_0001.mrg')[0]
>>> print text17
(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
    (ADJP (NP (CD 61) (NNS years)) (JJ old))
    (, ,))
  (VP
    (MD will)
    (VP
      (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director)))
      (NP-TMP (NNP Nov.) (CD 29))))
  (. .))
```

# POS Tagging & Parsing Complexity

- Uncommon usages of words

**The old man the boat**

```
>>> text18 = nltk.word_tokenize("The old man the boat")
>>> nltk.pos_tag(text19)
[('The', 'DT'), ('old', 'JJ'), ('man', 'NN'), ('the', 'DT'), ('boat', 'NN')]
```

- Well-formed sentences may still be meaningless!

**Colorless green ideas sleep furiously**

```
>>> text19 = nltk.word_tokenize("Colorless green ideas sleep furiously")
>>> nltk.pos_tag(text18)
[('Colorless', 'NNP'), ('green', 'JJ'), ('ideas', 'NNS'), ('sleep', 'VBP'),
('furiously', 'RB')]
```

# Take Home Concepts

- POS tagging provides insights into the word classes / types in a sentence
- Parsing the grammatical structures helps derive meaning
- Both tasks are difficult, linguistic ambiguity increases the difficulty even more
- Better models could be learned with supervised training
- NLTK provides access to tools and data for training