

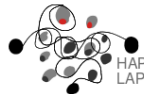
# Natural Language Toolkit (NLTK)

Olatz Perez de Vinaspre  
Ander Soraluze  
HAP/LAP.

*2021*

- Open source python modules and linguistic data for Natural Language Processing application
- Tokenization, Lemmatization, POS tagging, chunking, parsing...
- Several corpora: Brown, Penn Treebank corpus, Gutenberg...
- Developed by group of people, project leaders: Steven Bird, Edward Loper, Ewan Klein

# Preparing NLTK



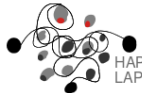
- We need data to experiment with:
  - `import nltk`
  - `nltk.download()`

# Basic commands



- Data from book:
  - `from nltk.book import *`
- See what is it
  - `print(text1)`
- Search the word:
  - `text1.concordance("monstrous")`
- Find similar words:
  - `text1.similar("monstrous")`

# Basic commands



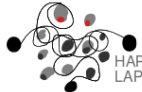
- Counting vocab
  - `len(text4)` gives tokens (words and punctuations)
- Vocabulary (unique words) used by author
  - `len(set(text1))`

# Basic commands



- See sorted vocabulary(upper case first!)
  - `sorted(set(text1))`
- Measuring richness of text
  - `len(text1)/len(set(text1))`
  - before division import for floating point division *from \_\_future\_\_  
import division*
- Counting occurrences
  - `text1.count("monstruous")`

# Frequency distribution



- Frequency distribution tells us the frequency of each vocabulary item in the text
- It is a distribution since it tells us how the total number of word token are distributed across the vocabulary

```
import nltk.probability
fdist = FreqDist(text1)
vocabulary = fdist.keys()
print(vocabulary[:50])
```

# Tokenization



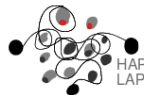
- Break up the string into words and punctuation

```
from nltk.tokenize import word_tokenize
sentence="There is a tree near the river."
tokens = nltk.word_tokenize(sentence)
text = nltk.Text(tokens)
print(tokens)
```

```
['There', 'is', 'a', 'tree', 'near', 'the', 'river', '.']
```



# Remove Stopwords



```
from nltk.corpus import stopwords
en_stops = set(stopwords.words('english'))
all_words = ['There', 'is', 'a',
             'tree', 'near', 'the', 'river', '.']
    for word in all_words:
        if word not in en_stops:
            print(word)
```

# Lemmatization



```
import nltk
from nltk.stem import WordNetLemmatizer
# Initialize the Wordnet Lemmatizer
wnl = WordNetLemmatizer()

# Lemmatize Single Word
print(wnl.lemmatize("bats"))
bat
print(lemmatizer.lemmatize("feet"))
foot

# Lemmatize list of words and join
lemmatized_output = ' '.join([wnl.lemmatize(w) for w
in word_list])
print(lemmatized_output)
```

```
print(nltk.pos_tag(['feet']))
```

```
[('feet', 'NNS')]
```

```
sentence="The striped bats are hanging on their feet  
for best"
```

```
print(nltk.pos_tag(nltk.word_tokenize(sentence)))
```

```
[('The', 'DT'), ('striped', 'JJ'), ('bats', 'NNS'),  
( 'are', 'VBP'), ('hanging', 'VBG'), ('on', 'IN'),  
( 'their', 'PRP$'), ('feet', 'NNS'), ('for', 'IN'),  
( 'best', 'JJS')]
```

- Gutenberg Corpus
  - Project Gutenberg electronic text archive
  - 25,000 free electronic books

```
import nltk
from nltk.corpus import gutenberg
# The file identifiers in this corpus
gutenberg.fileids()
# Select a text Emma by Jane Austen
emma = nltk.corpus.gutenberg.words('austen-emma.txt')
```

- Brown Corpus
  - First million-word electronic corpus of English
  - Contains text from 500 sources
  - The sources have been categorized by genre, such as news, editorial, and so on
  - The Brown Corpus is a convenient resource for studying systematic differences between genres

```
import nltk
from nltk.corpus import brown
# See the list of genres
brown.categories()
# Obtain words from files categorised as news
brown.words(categories='news')
# Obtain sents from files categorised as news
brown.sents(categories='news')
```

## Example using Brown corpus



```
import nltk
from nltk.corpus import brown
news_text = brown.words(categories='news')
fdist = nltk.FreqDist([w.lower() for w in news_text])
modals = ['can', 'could', 'may', 'might', 'must',
'will']
for m in modals:
    print m + ':', fdist[m]
```

# Accessing text from the web



```
from urllib import request
url = "http://www.gutenberg.org/files/2554/2554-0.txt"
response = request.urlopen(url)
raw = response.read().decode('utf8')
```

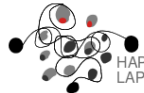
# Hypertext Markup Language(HTML)



- The vast majority of web pages are primarily represented in HTML
- It uses special codes representing additional information to elements in a document
- These codes primarily consist of tags
  - always enclosed between the two symbols < and >
  - normally come in pairs: opening and closing
  - specify the type of element being described



# HTML example



```
<!doctype html>
<html>
  <head>
    <title>This document now has a title </title>
  </head>
  <body>
    <h1>The largest header, useful for chapter titles</h1>
    <p>A paragraph where you can write all you want, in <i>italic</i> or <b>bold</b>.</p>
    <p>Another paragraph where you can write <br>
    all you want.</p>
  </body>
</html>
```

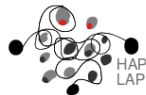
```
from urllib import request
url = "http://www.example.html"
response = request.urlopen(url)
raw = response.read().decode('utf8')
```

- It's HTML; Clean the tags...
- `raw = nltk.clean_html(html)`

```
nltk.clean_html(""" <p>This is some article text
with <a href='http://google.com'> a link to
Google</a></p>""")
'This is some article text with a link to Google'
```

Language processing task	NLTK modules	Functionality
Accessing corpora	<code>nltk.corpus</code>	Standardized interfaces to corpora and lexicons
String processing	<code>nltk.tokenize</code> , <code>nltk.stem</code>	Tokenizers, sentence tokenizers, stemmers
Collocation discovery	<code>nltk.collocations</code>	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	<code>nltk.tag</code>	n-gram, backoff, Brill, HMM, TnT
Classification	<code>nltk.classify</code> , <code>nltk.cluster</code>	Decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	<code>nltk.chunk</code>	Regular expression, n-gram, named entity
Parsing	<code>nltk.parse</code>	Chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	<code>nltk.sem</code> , <code>nltk.inference</code>	Lambda calculus, first-order logic, model checking
Evaluation metrics	<code>nltk.metrics</code>	Precision, recall, agreement coefficients
Probability and estimation	<code>nltk.probability</code>	Frequency distributions, smoothed probability distributions
Applications	<code>nltk.app</code> , <code>nltk.chat</code>	Graphical concordancer, parsers, WordNet browser, chatbots

# NLTK BOOK



<https://www.nltk.org/book/>