Natural Language Processing (NLP) - A hands-on introduction

Popular Libraries

- NLTK
- spaCy

NLTK & spaCy is a free open-source library for Natural Language Processing (NLP) in Python to support teaching, research, and development. Which are:-

- Free and Open source
- Easy to use
- Modular
- Well documented
- Simple and extensible

In this notebook, I will provide basic NLP tasks that we need in order to process raw text to find useful informations. For each tasks, we will be using NLTK as well as spaCy. Good news is that both are installed in Google Colab by default.

Some definitions

- **Corpus** - Corpora is the plural of Corpus. "**Corpus**" mainly appears in NLP area or application domain related to texts/documents, because of its meaning "a collection of written texts"
  - **Example:** A collection of news documents.
- **Dataset** - dataset appears in every application domain (in can be **image/video/text/numerical/mixed**) --- a collection of any kind of data is a dataset.
- **Lexicon** - vocabulary or list of Words and their meanings.
  - **Example:** English dictionary.
- **Token** - Each "entity" that is a part of whatever was split up based on rules.
  - For examples, each word is a token when a sentence is "tokenized" into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph.

Tokenization

Tokenization is the process of breaking a stream of text up into sentences, words, phrases, symbols, or other meaningful elements called tokens.

```
import nltk
nltk.download('punkt')

# For tokenizing words and sentences
from nltk.tokenize import word_tokenize, sent_tokenize

s = "Good muffins cost $3.88\nin New York. Please buy me two of them.\n\nThanks."

print (sent_tokenize(s))
print (word_tokenize(s))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['Good muffins cost $3.88\nin New York.', 'Please buy me two of them.', 'Thanks.']
['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York', '.', 'Please', 'buy', 'me'
```

```
import spacy

# Small spaCy model
nlp = spacy.load("en_core_web_sm")

doc = nlp("Good muffins cost $3.88\nin New York. Please buy me two of them.\n\nThanks.")

print("\n\nTokenized Sentences")

for i, sent in enumerate(doc.sents):
        print('-->Sentence %d: %s' % (i, sent.text))

print("\n\nTokenized Words")

tokens = [token.text for token in doc]
print(tokens)
```

```
/usr/local/lib/python3.8/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't ir
  warnings.warn("Can't initialize NVML")


Tokenized Sentences
-->Sentence 0: Good muffins cost $3.88
in New York.
-->Sentence 1: Please buy me two of them.


-->Sentence 2: Thanks.


Tokenized Words
```

```
['Good', 'muffins', 'cost', '$', '3.88', '\n', 'in', 'New', 'York', '.', 'Please', 'buy'
```

Downloading Large spaCy model

```
!python -m spacy download en_core_web_lg

import en_core_web_lg

nlp = en_core_web_lg.load()
```

```
/usr/local/lib/python3.8/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't ir
  warnings.warn("Can't initialize NVML")
2023-01-30 07:43:41.145527: E tensorflow/stream_executor/cuda/cuda_driver.cc:271] failed
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
Collecting en-core-web-lg==3.4.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_lg
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 587.7/587.7 MB 2.3 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.5.0,>=3.4.0 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: pathy>=0.3.5 in /usr/local/lib/python3.8/dist-packages (f
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.8/dis
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.8/d
Requirement already satisfied: thinc<8.2.0,>=8.1.0 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.8/dist-packages (
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: typer<0.8.0,>=0.3.0 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.8/dist-
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages
Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-packages (from sp
Requirement already satisfied: wasabi<1.1.0,>=0.9.1 in /usr/local/lib/python3.8/dist-pac
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-packages (fro
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.8/dist-
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.8/dist
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in /usr/local/lib/p
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.8/dist-
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.10 in /usr/local/lib/python3.8/d
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.8/dist
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.8/dist
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (f
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packag
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.8/dist
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.8/dist-package
Installing collected packages: en-core-web-lg
Successfully installed en-core-web-lg-3.4.1
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_lg')
```

Filtering stopwords

- **Stopwords** are common words that **generally** do not contribute to the meaning of a sentence.
- Most search engines will filter stopwords out of search queries and documents in order to **save space and time** in their index.
    - Removing stopwords is not a hard and fast rule in NLP. It depends upon the task that we are working on.
    - For tasks like text classification, where the text is to be classified into different categories, stopwords are removed or excluded from the given text so that more focus can be given to those words which define the meaning of the text.
- All Stopwords collection including Bengali.

```
nltk.download('stopwords')
from nltk.corpus import stopwords

# All english stopwords list
english_stops = set(stopwords.words('english'))

print (english_stops)

words = ['The', 'natural', 'language', 'processing', 'is', 'very', 'interesting']
filtered_words = [word for word in words if word.lower() not in english_stops]    # word.low

print(filtered_words)
```

```
{"haven't", 'where', 'out', 'the', 'mightn', 'down', 'shouldn', 'should', "hadn't", 'nee
['natural', 'language', 'processing', 'interesting']
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS

print('Number of stop words: %d' % len(spacy_stopwords))
print('First ten stop words: %s' % list(spacy_stopwords)[:10])
```

```
Number of stop words: 326
First ten stop words: ['hers', 'various', 'nobody', 'who', 'after', 'until', 'per', 'lea
```

```
doc = nlp("Good muffins cost $3.88\nin New York. Please buy me two of them.\n\nThanks.")

tokens = [token.text for token in doc if not token.is_stop]

print(tokens)
```

```
['Good', 'muffins', 'cost', '$', '3.88', '\n', 'New', 'York', '.', 'buy', '.', '\n\n', '
```

Adding Custom Stopwords

```
english_stops = set(stopwords.words('english'))

print (english_stops)

english_stops.remove('is')
english_stops.add('natural')


words = ['The', 'natural', 'language', 'processing', 'is', 'very', 'interesting']
filtered_words = [word for word in words if word.lower() not in english_stops]

print(filtered_words)
```

```
{'hers', 'wouldn', 'who', 'after', 'until', 'd', 'there', 'in', 'don', 'about', 'now', '
['language', 'processing', 'is', 'interesting']
```

Edit Distance

The edit distance is the number of character changes necessary to transform the given word into the suggested word.

```
from nltk.metrics import edit_distance

print(edit_distance("Birthday","Bday"))

print(edit_distance("university", "varsity"))
```

```
4
4
```

Removing Punctuation

```
import string
import nltk

nltk.download('punkt')

puncset = list(string.punctuation)

sentence = "Hun Sen's Cambodian can't People's Party won 64 of the 122 parliamentary seats i

sentence = sentence.lower()
print(sentence)
sentence = nltk.word_tokenize(sentence)
print(sentence)
sentence = [i for i in sentence if i not in puncset] # Removing punctuation
print(sentence)
sentence = [w for w in sentence if w.isalpha()] # Removing numbers and punctuation
print(sentence)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
hun sen's cambodian can't people's party won 64 of the 122 parliamentary seats in party
['hun', 'sen', "'s", 'cambodian', 'ca', "n't", 'people', "'s", 'party', 'won', '64', 'of
['hun', 'sen', "'s", 'cambodian', 'ca', "n't", 'people', "'s", 'party', 'won', '64', 'of
['hun', 'sen', 'cambodian', 'ca', 'people', 'party', 'won', 'of', 'the', 'parliamentary'
```

Normalizing Text

The goal of both stemming and lemmatization is to "**normalize**" words to their **common base form**, which is useful for many text-processing applications.

- **Stemming** = heuristically removing the affixes of a word, to get its **stem (root)**.
    - It is a rule-based process of stripping the suffixes **("ing", "ly", "es", "s" etc)** from a word
- **Lemmatization** = Lemmatization process involves first determining the part of speech of a word, and applying different normalization rules for each part of speech.

Consider:

- I was taking a **ride** in the car.
- I was **riding** in the car.

Imagine every word in the English language, every possible tense and affix you can put on a word. **Having individual dictionary entries per version would be highly redundant and inefficient.**

- Lisa **ate** the food and washed the dishes.
- They were **eating** noodles at a cafe.

- Don’t you want to **eat** before we leave?
- We have just **eaten** our breakfast.
- It also **eats** fruit and vegetables.

Unfortunately, that is not the case with machines. **They treat these words differently**. Therefore, we need to normalize them to their root word, which is **“eat”** in our example.

Stemming

- One of the **most popular** stemming algorithms is the Porter stemmer, which has been around since 1979.
- Several other stemming algorithms provided by NLTK are Lancaster Stemmer and Snowball Stemmer.

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

example_words = ["python","pythoner","pythoning","pythoned","pythonly"]

for w in example_words:
  print(stemmer.stem(w))
```

```
python
python
python
python
pythonli
```

Lemmatization

Lemmatize takes a part of speech parameter, "pos." **If not supplied, the default is "noun".**

```
## Lemmatization using NLTK

import nltk
from nltk.stem import WordNetLemmatizer
nltk.download("omw-1.4")

nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize('cooking'))
print(lemmatizer.lemmatize('cooking', pos='v'))  # noun = n, verb = v, ajdective = a
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package wordnet to /root/nltk_data...
cooking
cook
```

```
## Lemmatization using spaCy

doc = nlp('Jim bought 300 shares of Acme Corp. in 2006.')

lemma_words = []

for token in doc:
    lemma_words.append(token.lemma_)

print(lemma_words)
```

```
['Jim', 'buy', '300', 'share', 'of', 'Acme', 'Corp.', 'in', '2006', '.']
```

Comparison between stemming and lemmatizing

The major difference between these is, as you saw earlier, **stemming can often create non-existent words**, whereas **lemmas are actual words**, you can just look up in an English dictionary.

```
print(stemmer.stem('believes'))
print(lemmatizer.lemmatize('believes'))
```

```
believ
belief
```

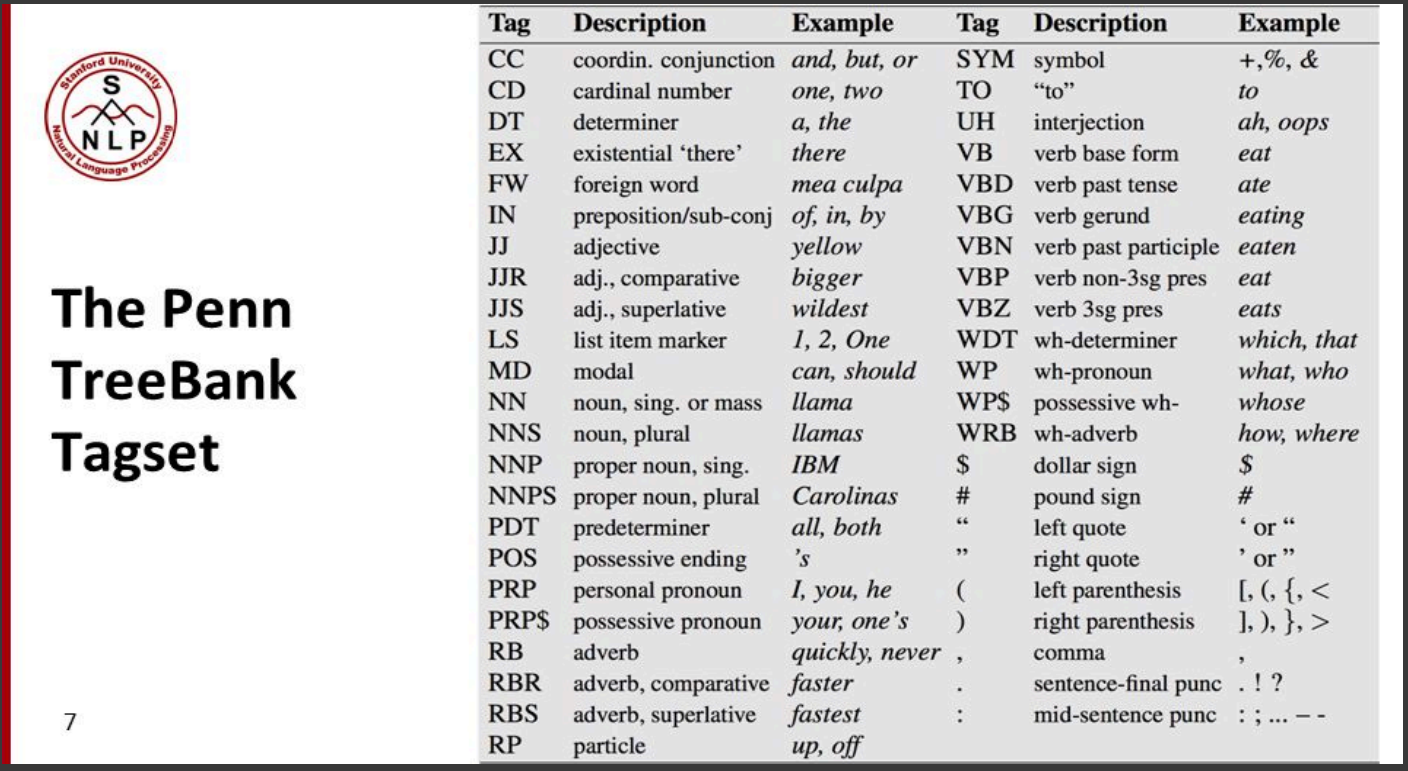
Part-of-speech Tagging

The English language is formed of different parts of speech (POS) like nouns, verbs, pronouns, adjectives, etc. POS tagging analyzes the words in a sentences and associates it with a POS tag

depending on the way it is used.

Full tag list.

Penn Bank Part-of-Speech Tags



Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

```
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag

nltk.download('averaged_perceptron_tagger')

words = word_tokenize('Jim bought 300 shares of Acme Corp. in 2006.')

tagged_words = pos_tag(words)

print(tagged_words)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[('Jim', 'NNP'), ('bought', 'VBD'), ('300', 'CD'), ('shares', 'NNS'), ('of', 'IN'), ('Ac
```

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp('Jim bought 300 shares of Acme Corp. in 2006.')

for token in doc:
    print(token.text, token.pos_, token.tag_)
```

```
Jim PROPN NNP
bought VERB VBD
300 NUM CD
shares NOUN NNS
of ADP IN
Acme PROPN NNP
Corp. PROPN NNP
in ADP IN
2006 NUM CD
. PUNCT .
```

Named-entity Recognition

Named-entity recognition is a subtask of information extraction that seeks to locate and classify elements in text into pre-defined categories such as the names of **persons**, **organizations**, **locations**, **expressions of times**, **quantities**, **monetary values**, **percentages**, etc.

NE Type and Examples:-

- **ORGANIZATION** - Georgia-Pacific Corp., WHO
- **PERSON** - Eddy Bonte, President Obama
- **LOCATION** - Murray River, Mount Everest
- **DATE**- June, 2008-06-29
- **TIME** - two fifty a m, 1:30 p.m.
- **MONEY** - 175 million Canadian Dollars, GBP 10.40
- **PERCENT** - twenty pct, 18.75 %
- **FACILITY** - Washington Monument, Stonehenge
- **GPE** - South East Asia, Midlothian

```
from nltk import pos_tag, ne_chunk
from nltk.tokenize import wordpunct_tokenize

nltk.download('maxent_ne_chunker')
nltk.download('words')

sent = 'Jim bought 300 shares of Acme Corp. in 2006.'

print(ne_chunk(pos_tag(wordpunct_tokenize(sent))))
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
(S
  (PERSON Jim/NNP)
  bought/VBD
  300/CD
  shares/NNS
  of/IN
  (ORGANIZATION Acme/NNP Corp/NNP)
  ./.
  in/IN
  2006/CD
  ./.)
```

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Jim bought 300 shares of Acme Corp. in 2006.")

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

```
Jim 0 3 PERSON
300 11 14 CARDINAL
Acme Corp. 25 35 ORG
2006 39 43 DATE
```

```
Start coding or generate with AI.
```