Overview and usage of NLTK, comparison with spaCy

Introduction

Natural language processing (NLP) is a field that focuses on making natural human language usable by computer programs. The increment in the usage of Social Media has grown the size of text data, and boost the studies or researches in Natural Language Processing (NLP). All the smart and voice activated devices uses some form of NLP application. NLTK (Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. In this article we will go over general functionalities provided by NLTK and comparison with spaCy which is another python package for NLP.

Body

NLTK provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers. I have tried few of the above functionalities as part of this review.
#TOKENISATION

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

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/niveditachatterjee/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.

True
```

```
ION -Splitting bigger parts to small parts. We can tokenize paragraphs to sentences and sentences to words.
: = """NLTK has been called "a wonderful tool for teaching, and working in, computational linguistics using Python," and
```

```
tokened_sent = nl.sent_tokenize(sample_text)
tokened_word = nl.word_tokenize(sample_text)
print(tokened_sent)
print(tokened_word)
```

```
['NLTK has been called "a wonderful tool for teaching, and working in, computational linguistics using Python," and
"an amazing library to play with natural language."']
['NLTK', 'has', 'been', 'called', '``', 'a', 'wonderful', 'tool', 'for', 'teaching', ',', 'and', 'working', 'in',
',', 'computational', 'linguistics', 'using', 'Python', ',', "''", 'and', '``', 'an', 'amazing', 'library', 'to', 'pl
ay', 'with', 'natural', 'language', '.', "''"]
```

#Stemming

```
#STEMMING- Removing affixes from words and returning the root word
ps = nl.PorterStemmer()
for w in tokened_word:
    print(ps.stem(w))
```

```
nltk
ha
been
call
``
a
wonder
tool
for
teach
,
and
work
in
,
comput
linguist
use
python
,
''
and
``
an
amaz
librari
to
play
with
natur
```

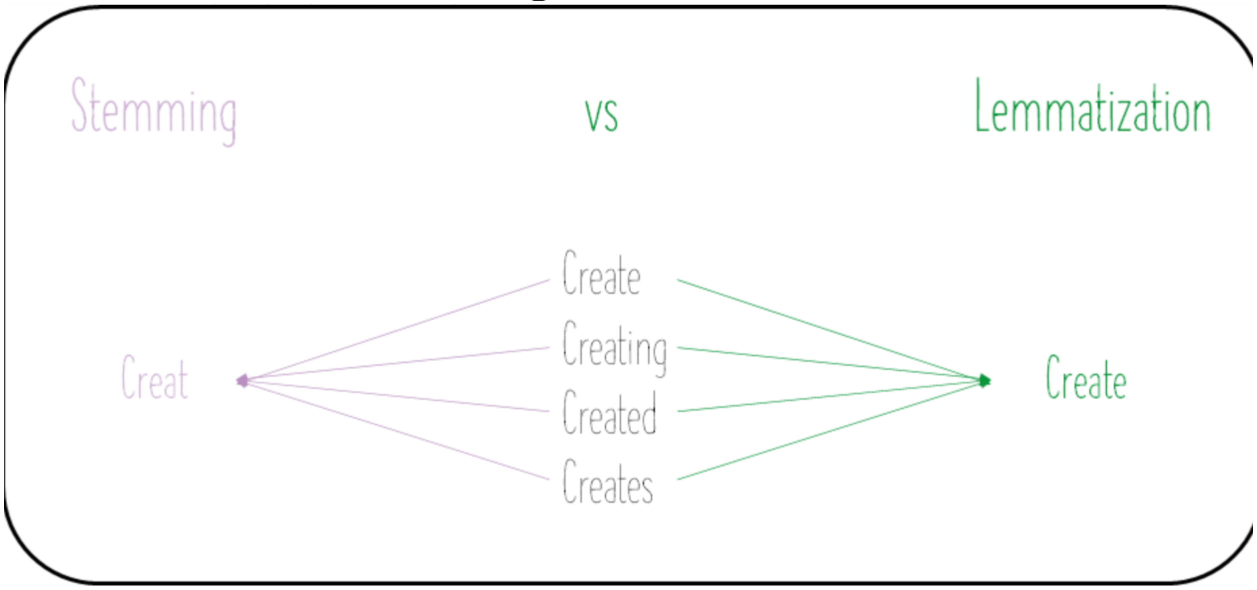
#Lemmatization

```
#Lemmatization - Word lemmatizing is similar to stemming, but the difference lies in the output. The Lemmatized output

from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize('increases'))
print(lemmatizer.lemmatize('playing', pos="v"))
print(lemmatizer.lemmatize('playing', pos="v"))
print(lemmatizer.lemmatize('playing', pos="n"))
print(lemmatizer.lemmatize('playing', pos="a"))
print(lemmatizer.lemmatize('playing', pos="r"))
print(lemmatizer.lemmatize("cats"))
print(lemmatizer.lemmatize("cacti"))
print(lemmatizer.lemmatize("geese"))
print(lemmatizer.lemmatize("rocks"))
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/niveditachatterjee/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

```
increase
play
play
playing
playing
playing
cat
cactus
goose
rock
```

Difference between Stemming and Lemmatization



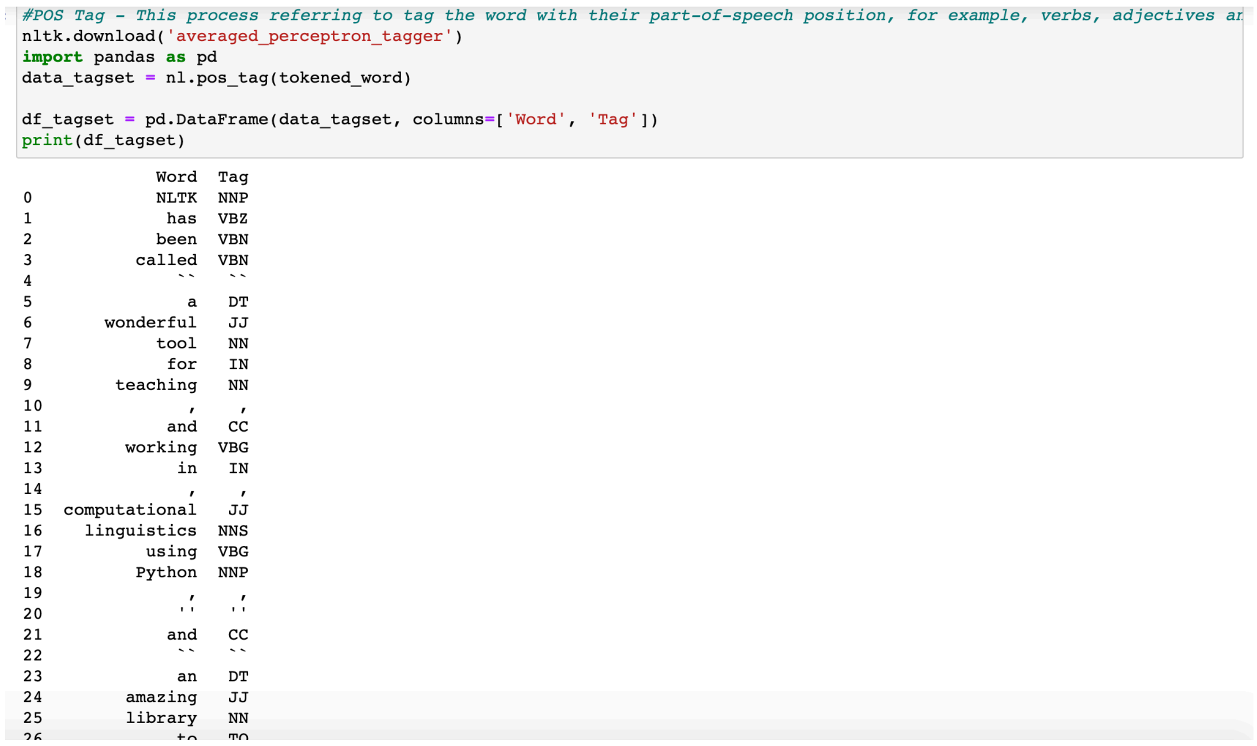
#Removing Stop words

```
#Stop words: There are some words in English like “the,” “of,” “a,” “an,” and so on. These are ‘stop words’. Stop words
nl.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
example_sent = "This is a sample sentence, showing off the stop words filtration. Here you can write whatever you want
#STOP WORDS ARE PARTICULAR TO RESPECTIVE LANGUAGES(english, spanish, french Et cetera)
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(example_sent)
filtered_sentence = [w for w in word_tokens if w not in stop_words]
print(filtered_sentence)
```

```
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.', 'Here', 'write', 'whatever', 'wan
t', '.', 'You', 'also', 'add', 'big', 'text', 'file', 'see', 'technique', 'works']
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/niveditachatterjee/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

#POS Tag



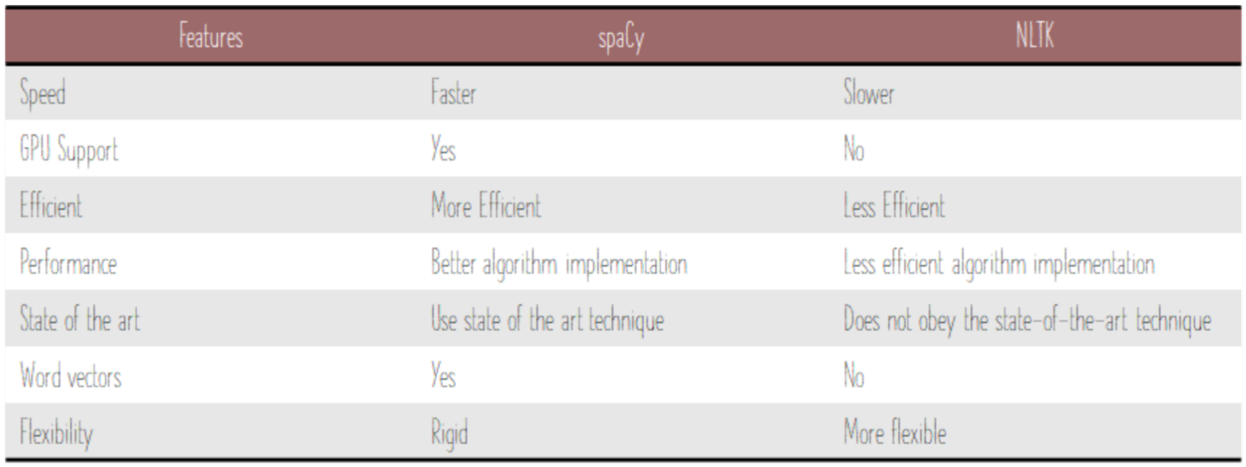
```
#POS Tag - This process referring to tag the word with their part-of-speech position, for example, verbs, adjectives ar
nltk.download('averaged_perceptron_tagger')
import pandas as pd
data_tagset = nl.pos_tag(tokened_word)

df_tagset = pd.DataFrame(data_tagset, columns=['Word', 'Tag'])
print(df_tagset)
```

```
             Word  Tag
0            NLTK  NNP
1             has  VBZ
2            been  VBN
3          called  VBN
4              ``   ``
5               a   DT
6       wonderful   JJ
7            tool   NN
8             for   IN
9        teaching   NN
10              ,    ,
11            and   CC
12        working  VBG
13             in   IN
14              ,    ,
15  computational   JJ
16    linguistics  NNS
17          using  VBG
18         Python  NNP
19              ,    ,
20             ''   ''
21            and   CC
22             ``   ``
23             an   DT
24        amazing   JJ
25        library   NN
26             to   TO
```

In this section we will discuss the major differences between NLTK and spaCy

- A major difference lies how these libraries were built.
  - NLTK is essentially a string processing library, where each function takes strings as input and returns a processed string
  - spaCy takes an object-oriented approach. Each function returns objects instead of strings or arrays. This allows for easy exploration of the tool.
- Each library utilizes either time or space to improve performance. While NLTK returns results much slower than spaCy. spaCy consumes a lot of memory.spaCy is written in cython which makes it much faster than NLTK
- Advantage of SpaCy compared to NLTK - it simplified the text processing.

    - Ex: In spaCy, user may obtain the POS Tag and the lemma (root form of the word) when you are performing tokenization, which saves some effort.
- In NLTK documentation we can find below statement which supports the theory that it is a suitable library for research purpose and play around NLP .
  NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”
- In Contrast spaCy markets itself as “Industrial-strength Natural Language Processing”. It is “Blazing fast”, “get things done”,” awesome ecosystem”.
- Visual comparison of spaCy and NLTK



Features	spaCy	NLTK
Speed	Faster	Slower
GPU Support	Yes	No
Efficient	More Efficient	Less Efficient
Performance	Better algorithm implementation	Less efficient algorithm implementation
State of the art	Use state of the art technique	Does not obey the state-of-the-art technique
Word vectors	Yes	No
Flexibility	Rigid	More flexible

Conclusion

Both NLTK and spaCy are very good option to build NLP solution. Depending on the use case we can choose which one to use. For a production system spaCy will be a better option because of processing speed and user friendliness. As part of this article, I read about both and tried few uses of NLTK. I will focus on learning spaCy because this is better suited for production jobs and “Get things done”.

References

https://www.activestate.com/blog/natural-language-processing-nltk-vs-spacy/
https://www.nltk.org/
https://spacy.io/
https://github.com/explosion/spaCy
https://github.com/nltk/nltk/wiki
https://towardsdatascience.com/text-processing-in-python-29e86ea4114c
https://towardsdatascience.com/text-normalization-with-spacy-and-nltk-1302ff430119