**Explore Natural Language Processing using Spacy Library**

**Introduction:**

Natural Language Processing is the process of interpretation or manipulation of human language in any form by a software or a machine in general sense. There are many libraries that can be used for processing human language namely NLTK, Spacy, Textblob, Genism.

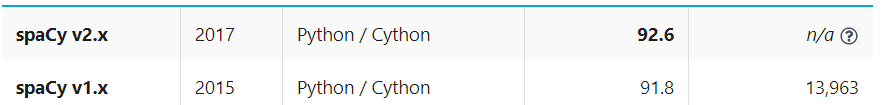
For this study we will be seeing some of the features of Spacy library extensively via a python experiment and a walk through.

**Spacy:**

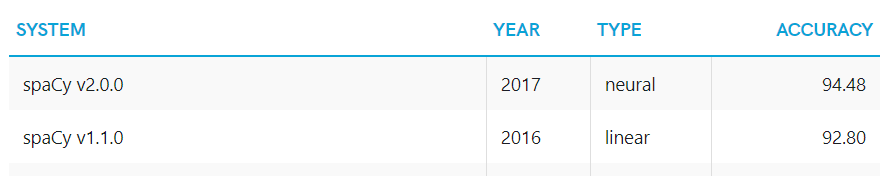
This is a library, that unlike NLTK doesn’t bombard us with too many options. As a result of which doesn’t waste much of the user’s time. It kind of gives only the best algorithm for the purpose at hand.

Spacy is the best pick if speed of NLP processing is under important considerations for your application. Its written in Cython which makes it faster. Spacy is the fastest in the world for the job of NLP interpretations. Cython is a static compiler which is used for writing c extensions for python.

**Benchmarks**



**Algorithm comparison chart:**



Aditya Prakash V- Exploring Spacy as an Alternative to NLTK

In [2]:

from spacy.lang.en import English

#from spacy.en import English

In [3]:

# Load English tokenizer, tagger, parser, NER and word vectors

nlp = English()

#This library contains the corpus of proper english words unlike the myriad of corpuses available in NLTK.

In [4]:

text = """He determined to drop his litigation with the monastry, and relinguish his claims to the wood-cuting and

fishery rihgts at once. He was the more ready to do this becuase the rights had become much less valuable, and he had

indeed the vaguest idea where the wood and river in question were."""

In [5]:

# "nlp" Object is used to create documents with linguistic annotations.

my\_doc = nlp(text)

In [6]:

# Create list of word tokens

token\_list = []

for token in my\_doc:

token\_list.append(token.text)

In [7]:

from spacy.lang.en.stop\_words import STOP\_WORDS

# Create list of word tokens after removing stopwords

filtered\_sentence =[]

In [8]:

for word in token\_list:

lexeme = nlp.vocab[word]

if lexeme.is\_stop == False:

filtered\_sentence.append(word)

In [9]:

print(token\_list)

['He', 'determined', 'to', 'drop', 'his', 'litigation', 'with', 'the', 'monastry', ',', 'and', 'relinguish', 'his', 'claims', 'to', 'the', 'wood', '-', 'cuting', 'and', '\n', 'fishery', 'rihgts', 'at', 'once', '.', 'He', 'was', 'the', 'more', 'ready', 'to', 'do', 'this', 'becuase', 'the', 'rights', 'had', 'become', 'much', 'less', 'valuable', ',', 'and', 'he', 'had', '\n', 'indeed', 'the', 'vaguest', 'idea', 'where', 'the', 'wood', 'and', 'river', 'in', 'question', 'were', '.']

In [10]:

print(filtered\_sentence)

['determined', 'drop', 'litigation', 'monastry', ',', 'relinguish', 'claims', 'wood', '-', 'cuting', '\n', 'fishery', 'rihgts', '.', 'ready', 'becuase', 'rights', 'valuable', ',', '\n', 'vaguest', 'idea', 'wood', 'river', 'question', '.']

Text Normalization using spaCy

In [15]:

import en\_core\_web\_sm

nlp = en\_core\_web\_sm.load()

doc = nlp(u"""He determined to drop his litigation with the monastry, and relinguish his claims to the wood-cuting and

fishery rihgts at once. He was the more ready to do this becuase the rights had become much less valuable, and he had

indeed the vaguest idea where the wood and river in question were.""")

lemma\_word1 = []

for token in doc:

lemma\_word1.append(token.lemma\_)

lemma\_word1

# This is the lemmatisation of all the pronouns with the pronouns in the text labelled as pronouns.

Out[15]:

['-PRON-',

'determine',

'to',

'drop',

'-PRON-',

'litigation',

'with',

'the',

'monastry',

',',

'and',

'relinguish',

'-PRON-',

'claim',

'to',

'the',

'wood',

'-',

'cut',

'and',

'\n',

'fishery',

'rihgts',

'at',

'once',

'.',

'-PRON-',

'be',

'the',

'more',

'ready',

'to',

'do',

'this',

'becuase',

'the',

'right',

'have',

'become',

'much',

'less',

'valuable',

',',

'and',

'-PRON-',

'have',

'\n',

'indeed',

'the',

'vague',

'idea',

'where',

'the',

'wood',

'and',

'river',

'in',

'question',

'be',

'.']

Parts of Sentences Analysis

In [14]:

import en\_core\_web\_sm

nlp = en\_core\_web\_sm.load()

# Process whole documents

text = ("When Sebastian Thrun started working on self-driving cars at "

"Google in 2007, few people outside of the company took him "

"seriously. “I can tell you very senior CEOs of major American "

"car companies would shake my hand and turn away because I wasn’t "

"worth talking to,” said Thrun, in an interview with Recode earlier "

"this week.")

doc = nlp(text)

# Analyze syntax

print("Noun phrases:", [chunk.text for chunk in doc.noun\_chunks])

print("Verbs:", [token.lemma\_ for token in doc if token.pos\_ == "VERB"])

# Find named entities, phrases and concepts

for entity in doc.ents:

print(entity.text, entity.label\_)

#As we see below this is like worlds apart from NLTK as this gives the prefect workout of an nlp sentence as noun phrase and verb phrase which is not an available feaure in NLTK.

Noun phrases: ['Sebastian Thrun', 'self-driving cars', 'Google', 'few people', 'the company', 'him', 'I', 'you', 'very senior CEOs', 'major American car companies', 'my hand', 'I', 'Thrun', 'an interview', 'Recode']

Verbs: ['start', 'work', 'drive', 'take', 'can', 'tell', 'would', 'shake', 'turn', 'talk', 'say']

Sebastian Thrun PERSON

Google ORG

2007 DATE

American NORP

Thrun ORG

earlier this week DATE

POS tagging

In [13]:

import spacy

nlp = en\_core\_web\_sm.load()

doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

for token in doc:

print(token.text, token.lemma\_, token.pos\_, token.tag\_, token.dep\_,

token.shape\_, token.is\_alpha, token.is\_stop)

# The below is the breakup of the sentence into its gramatical parts word by word which cannot be expected in NLTK.

Apple Apple PROPN NNP nsubj Xxxxx True False

is be AUX VBZ aux xx True True

looking look VERB VBG ROOT xxxx True False

at at ADP IN prep xx True True

buying buy VERB VBG pcomp xxxx True False

U.K. U.K. PROPN NNP compound X.X. False False

startup startup NOUN NN dobj xxxx True False

for for ADP IN prep xxx True True

$ $ SYM $ quantmod $ False False

1 1 NUM CD compound d False False

billion billion NUM CD pobj xxxx True False

In [ ]:

**References:**

Orman, Treude 2017. **Choosing an NLP Library for Analyzing Software Documentation: A Systematic Literature Review and a Series of Experiments**, 187-197.

Schmitt, Kubler, Robert, Papdakis, LeTron 2019. **A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate**, 338-343.

Partalidou, Spyromitros-Xioufis, Vologiannidis, Doropoulos, Diamantaras 2020. **Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy** 337-341.

Spacy Documentation- <https://spacy.io/>

Cython Documentation- <https://cython.org/>

Analytics Vidyalaya - [https://www.analyticsvidhya.com/blog/2019/08/how-to-remove-stopwords-text-normalization-nltk-spacy-gensim-python/?#](https://www.analyticsvidhya.com/blog/2019/08/how-to-remove-stopwords-text-normalization-nltk-spacy-gensim-python/?)

Elite Data Science- <https://elitedatascience.com/python-nlp-librariesi>