NLP Essentials: Removing Stopwords and Performing Text Normalization using NLTK and spaCy in Python

INTERMEDIATE NLP PYTHON TECHNIQUE TEXT UNSTRUCTURED DATA

Overview

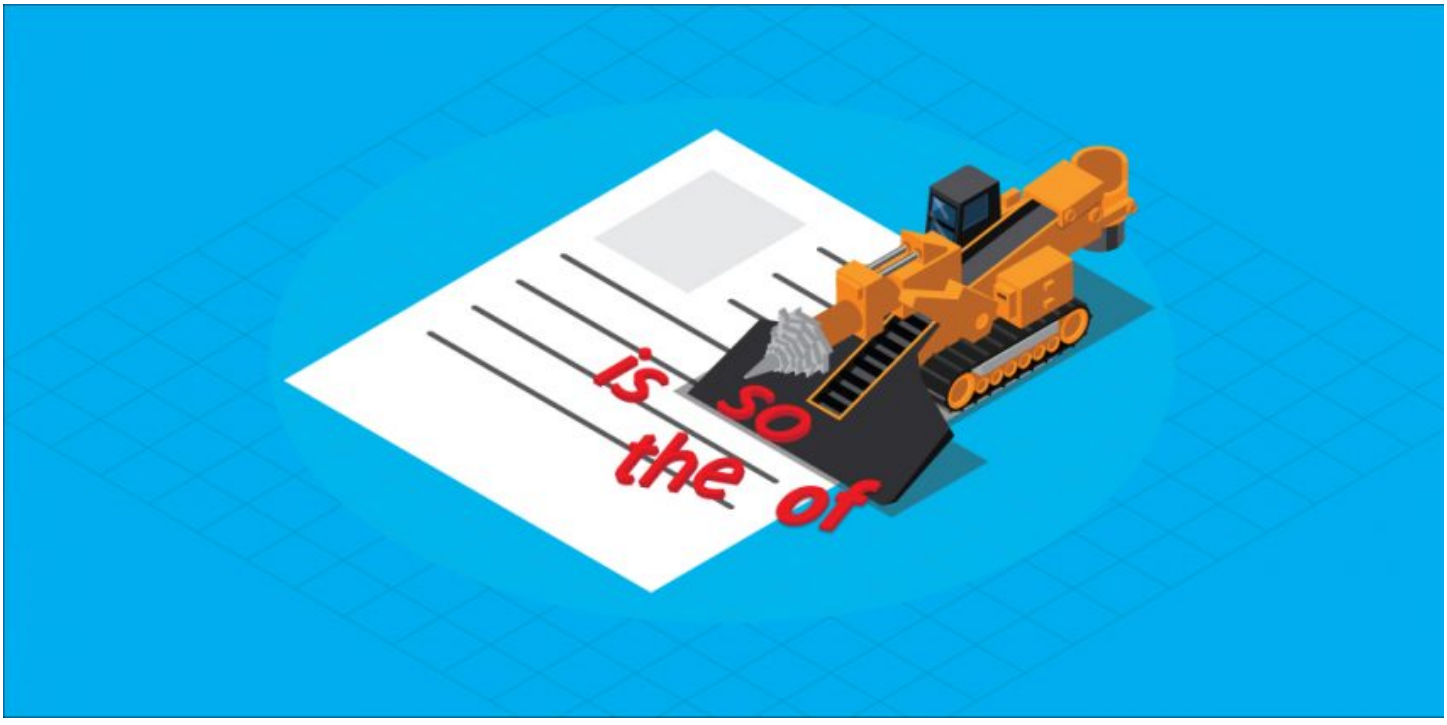
- Learn how to remove stopwords and perform text normalization in Python – an essential Natural Language Processing (NLP) read
- We will explore the different methods to remove stopwords as well as talk about text normalization techniques like stemming and lemmatization
- Put your theory into practice by performing stopwords removal and text normalization in Python using the popular NLTK, spaCy and Gensim libraries

Introduction

Don't you love how wonderfully diverse Natural Language Processing (NLP) is? Things we never imagined possible before are now just a few lines of code away. It's delightful!

But working with text data brings its own box of challenges. Machines have an almighty struggle dealing with raw text. We need to perform certain steps, called preprocessing, before we can work with text data using NLP techniques.

Miss out on these steps, and we are in for a botched model. These are essential NLP techniques you need to incorporate in your code, your framework, and your project.



We discussed the first step on how to get started with NLP in this article. Let's take things a little further and take a leap. We will discuss how to remove stopwords and perform text normalization in Python using a few very popular NLP libraries – NLTK, spaCy, Gensim, and TextBlob.

Are you a beginner in NLP? Or want to get started with machine learning but aren't sure where to begin? We have these two fields comprehensively covered in our end-to-end courses:

- *Natural Language Processing (NLP) Using Python*
- *Applied Machine Learning – Beginner to Professional*

Table of Contents

What are Stopwords?

Stopwords are the most common words in any natural language. For the purpose of analyzing text data and building NLP models, these stopwords might not add much value to the meaning of the document.

> Generally, the most common words used in a text are "the", "is", "in", "for", "where", "when", "to", "at" etc.

Consider this text string – "There is a pen on the table". Now, the words "is", "a", "on", and "the" add no meaning to the statement while parsing it. Whereas words like "there", "book", and "table" are the keywords and tell us what the statement is all about.

A note here – we need to perform tokenization before removing any stopwords. I encourage you to go through my article below on the different methods to perform tokenization:

- How to Get Started with NLP – 6 Unique Methods to Perform Tokenization

Here's a basic list of stopwords you might find helpful:

```
a about after all also always am an and any are at be been being but by came can cant come could did didn't do does doesn't doing don't else for from get give goes going had happen has have having how i if ill i'm in into is isn't it its i've just keep let like made make many may me mean more most much no not now of only or our really say see some something take tell than that the their them then they thing this to try up us use used uses very want was way we what when where which who why will with without wont you your youre
```

Why do we Need to Remove Stopwords?

Quite an important question and one you must have in mind.

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.

Just like we saw in the above section, words like **there**, **book**, and table add more meaning to the text as compared to the words **is** and **on**.

However, in tasks like machine translation and text summarization, removing stopwords is not advisable.

Here are a few key benefits of removing stopwords:

- On removing stopwords, dataset size decreases and the time to train the model also decreases
- Removing stopwords can potentially help improve the performance as there are fewer and only meaningful tokens left. Thus, it could increase classification accuracy
- Even search engines like Google remove stopwords for fast and relevant retrieval of data from the database

When Should we Remove Stopwords?

I've summarized this into two parts: when we can remove stopwords and when we should avoid doing so.

Remove Stopwords

We can remove stopwords while performing the following tasks:

- Text Classification
  - Spam Filtering
  - Language Classification
  - Genre Classification
- Caption Generation
- Auto-Tag Generation

Avoid Stopword Removal

- Machine Translation
- Language Modeling
- Text Summarization
- Question-Answering problems

Feel free to add more NLP tasks to this list!

Different Methods to Remove Stopwords

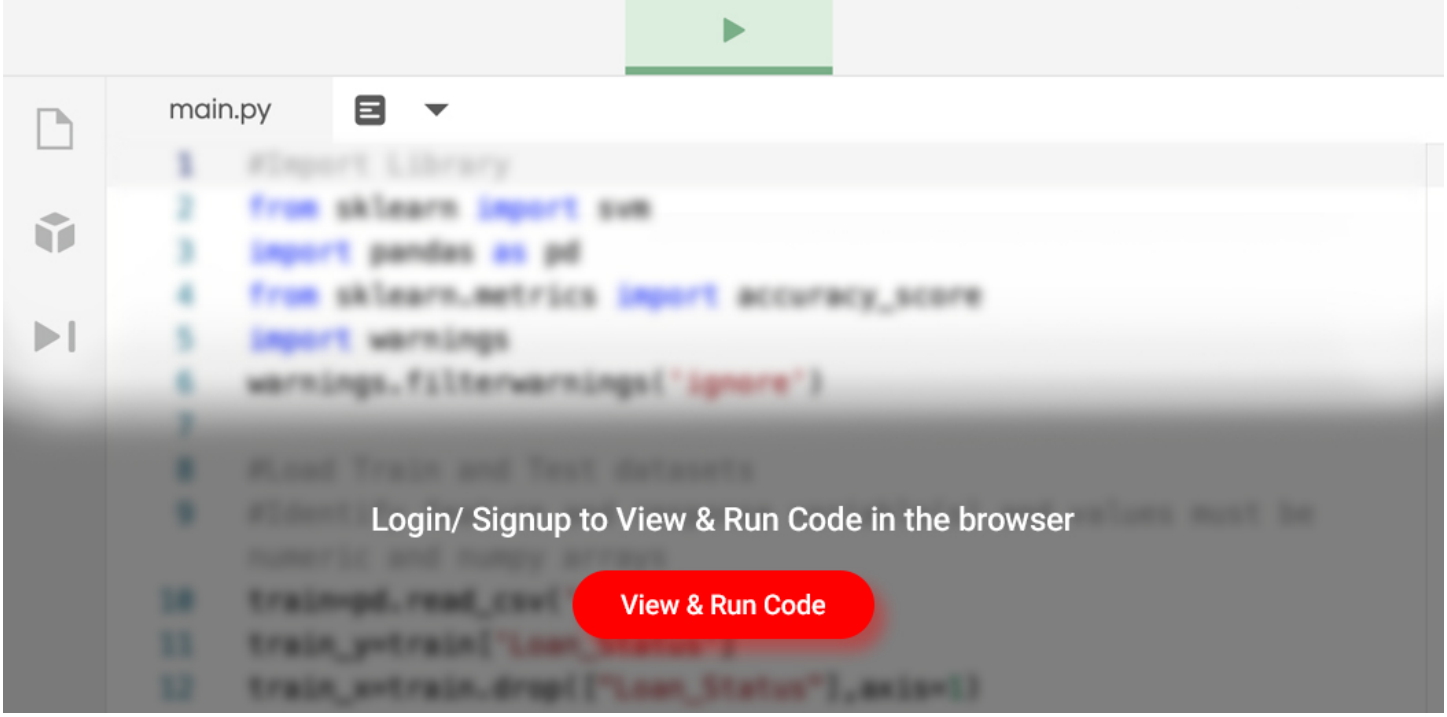
1. Stopword Removal using NLTK

NLTK, or the Natural Language Toolkit, is a treasure trove of a library for text preprocessing. It's one of my favorite Python libraries. **NLTK has a list of stopwords stored in 16 different languages.**

You can use the below code to see the list of stopwords in NLTK:

```
import nltk from nltk.corpus import stopwords set(stopwords.words('english'))
```

Now, to remove stopwords using NLTK, you can use the following code block. This is a LIVE coding window so you can play around with the code and see the results without leaving the article!



Here is the list we obtained after tokenization:

```
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.
```

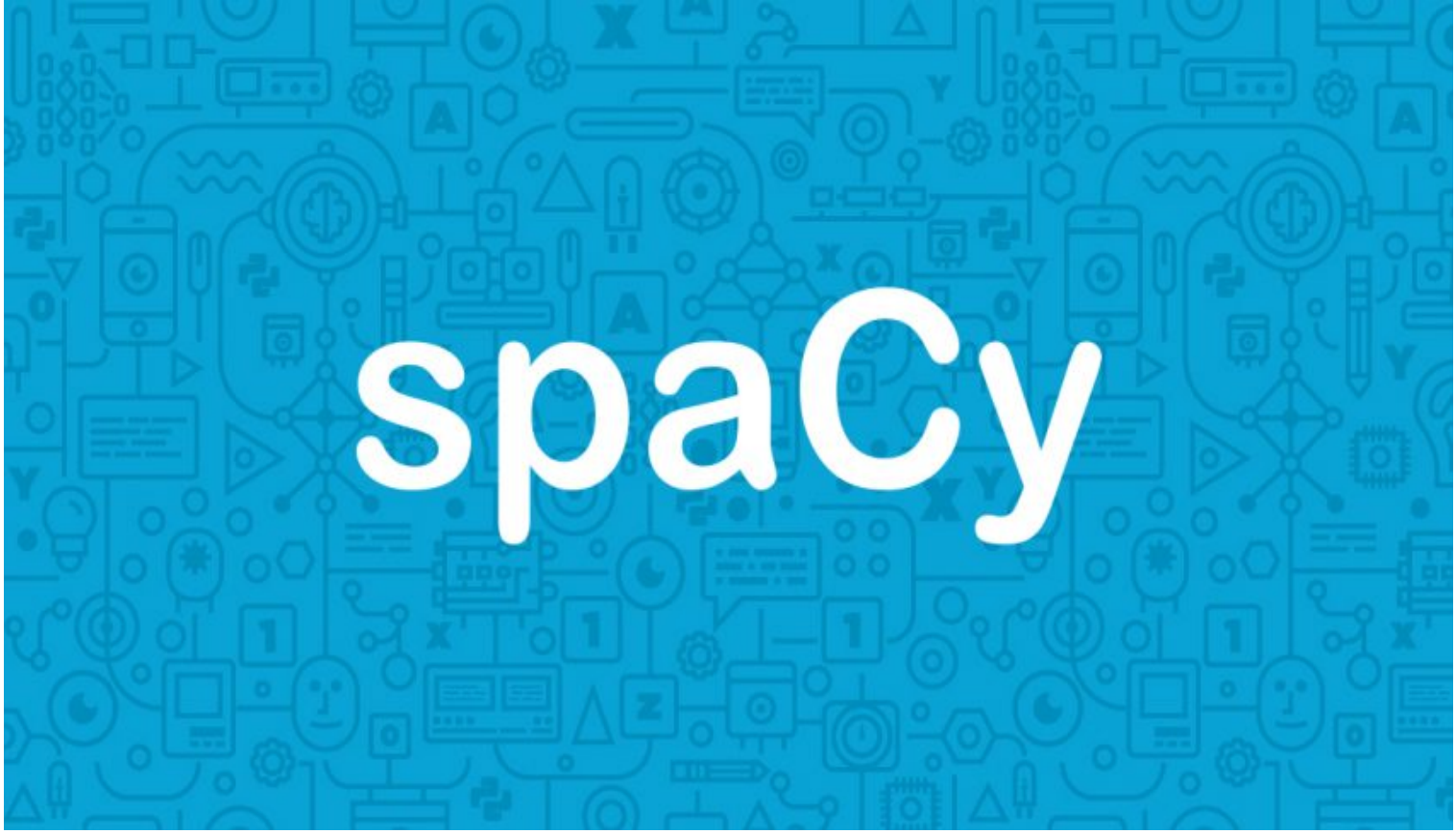
And the list after removing stopwords:

```
He determined drop litigation monastry, relinguish claims wood-cuting fishery rihgts. He ready becuase rights become much less valuable, indeed vaguest idea wood river question.
```

Notice that the size of the text has almost reduced to half! Can you visualize the sheer usefulness of removing stopwords?

2. Stopword Removal using spaCy

spaCy is one of the most versatile and widely used libraries in NLP. We can quickly and efficiently remove stopwords from the given text using SpaCy. It has a list of its own stopwords that can be imported as **STOP_WORDS** from the **spacy.lang.en.stop_words** class.



Here's how you can remove stopwords using spaCy in Python:

```
from spacy.lang.en import English

# Load English tokenizer, tagger, parser, NER and word vectors
nlp = English()

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."""

#  "nlp" Object is used to create documents with linguistic annotations.
my_doc = nlp(text)

# Create list of word tokens
token_list = []
for token in my_doc:
    token_list.append(token.text)

from spacy.lang.en.stop_words import STOP_WORDS

# Create list of word tokens after removing stopwords
filtered_sentence =[]

for word in token_list:
    lexeme = nlp.vocab[word]
    if lexeme.is_stop == False:
        filtered_sentence.append(word)
print(token_list)
print(filtered_sentence)
```

view raw

spacy.py hosted with ❤ by **GitHub**

This is the list we obtained after tokenization:

```
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.
```

And the list after removing stopwords:

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

An important point to note – stopword removal doesn't take off the punctuation marks or newline characters. We will need to remove them manually.

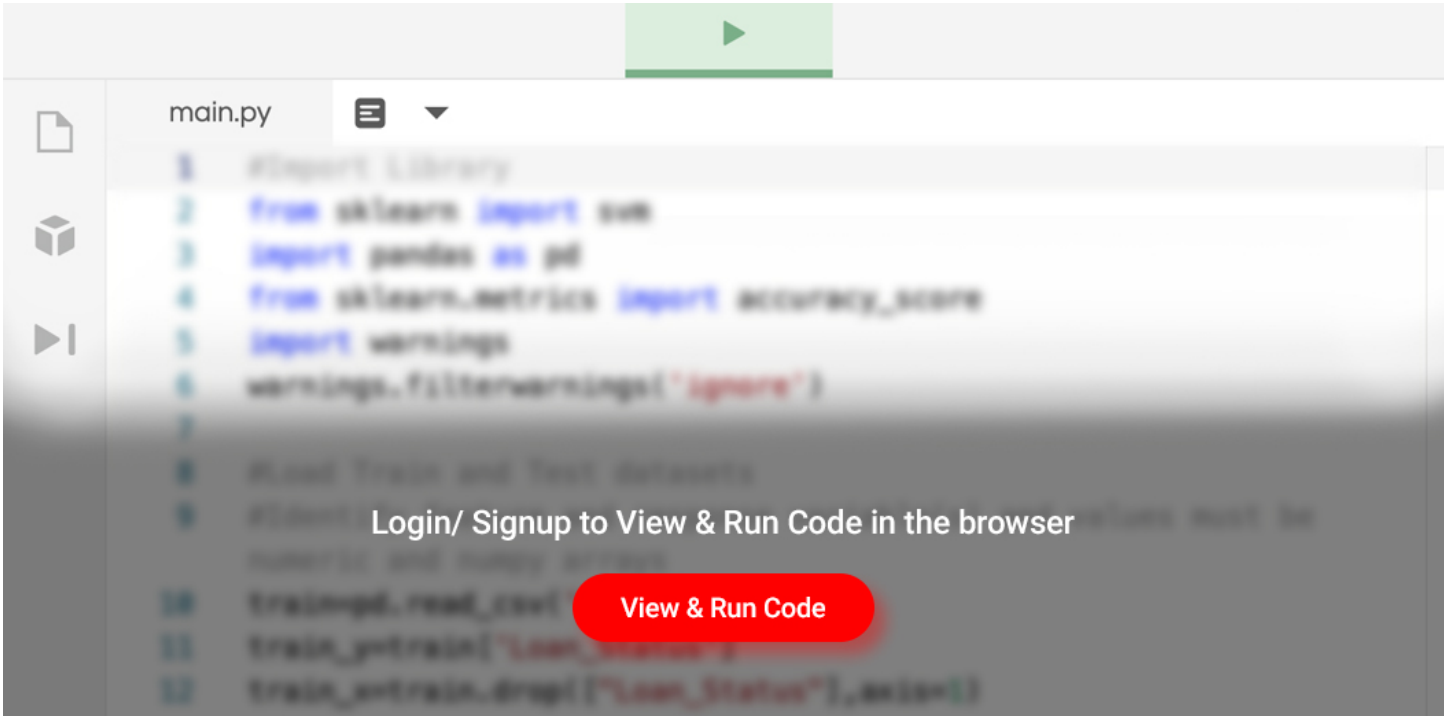
Read more about spaCy in this article with the library's co-founders:

- [DataHack Radio #23: Ines Montani and Matthew Honnibal – The Brains behind spaCy]

3. Stopword Removal using Gensim

Gensim is a pretty handy library to work with on NLP tasks. While pre-processing, gensim provides methods to remove stopwords as well. We can easily import the **remove_stopwords** method from the class **gensim.parsing.preprocessing.**

Try your hand on Gensim to remove stopwords in the below live coding window:




```
He determined drop litigation monastry, relinguish claims wood-cuting fishery rihgts once. He ready becuase rights valuable, vaguest idea wood river question were.
```

While using gensim for removing stopwords, we can directly use it on the raw text. There's no need to perform tokenization before removing stopwords. This can save us a lot of time.

Introduction to Text Normalization

In any natural language, words can be written or spoken in more than one form depending on the situation. That's what makes the language such a thrilling part of our lives, right? For example:

- 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.

In all these sentences, we can see that the word **eat** has been used in multiple forms. For us, it is easy to understand that eating is the activity here. So it doesn't really matter to us whether it is 'ate', 'eat', or 'eaten' – we know what is going on.

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.

Hence, **text normalization** is a process of transforming a word into a single canonical form. This can be done by two processes, **stemming** and **lemmatization**. Let's understand what they are in detail.

What are Stemming and Lemmatization?

> Stemming and Lemmatization is simply normalization of words, which means reducing a word to its root form.

In most natural languages, a root word can have many variants. For example, the word 'play' can be used as 'playing', 'played', 'plays', etc. You can think of similar examples (and there are plenty).

Stemming

Let’s first understand stemming:

- Stemming is a text normalization technique that cuts off the end or beginning of a word by taking into account a list of common prefixes or suffixes that could be found in that word
- It is a rudimentary rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word

Lemmatization

Lemmatization, on the other hand, is an organized & step-by-step procedure of obtaining the root form of the word. It makes use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations).

Why do we need to Perform Stemming or Lemmatization?

Let’s consider the following two sentences:

- **He was driving**
- **He went for a drive**

We can easily state that both the sentences are conveying the same meaning, that is, driving activity in the past. A machine will treat both sentences differently. Thus, to make the text understandable for the machine, we need to perform stemming or lemmatization.

Another benefit of text normalization is that it reduces the number of unique words in the text data. This helps in bringing down the training time of the machine learning model (and don’t we all want that?).

S0, which one should we prefer?

Stemming algorithm works by cutting the suffix or prefix from the word. **Lemmatization** is a more powerful operation as it takes into consideration the morphological analysis of the word.

Lemmatization returns the lemma, which is the root word of all its inflection forms.

We can say that stemming is a quick and dirty method of chopping off words to its root form while on the other hand, lemmatization is an intelligent operation that uses dictionaries which are created by in-depth linguistic knowledge. **Hence, Lemmatization helps in forming better features.**

Methods to Perform Text Normalization

1. Text Normalization using NLTK

The NLTK library has a lot of amazing methods to perform different steps of data preprocessing. There are methods like *PorterStemmer()* and *WordNetLemmatizer()* to perform stemming and lemmatization, respectively.

Let's see them in action.

Stemming

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

set(stopwords.words('english'))

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."""

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(text)

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

Stem_words = []
ps =PorterStemmer()
for w in filtered_sentence:
    rootWord=ps.stem(w)
    Stem_words.append(rootWord)
print(filtered_sentence)
print(Stem_words)
```

view raw

nltkstem.py hosted with ❤ by GitHub

```
He determined drop litigation monastry, relinguish claims wood-cuting fishery rihgts. He ready becuase rights become much less valuable, indeed vaguest idea wood river question.
```

```
He determin drop litig monastri, relinguish claim wood-cut fisheri rihgt. He readi becuas right become much less valuabl, inde vaguest idea wood river question.
```

We can clearly see the difference here. Now, let's perform lemmatization on the same text.

Lemmatization

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
from nltk.stem import WordNetLemmatizer
set(stopwords.words('english'))

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."""

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(text)

filtered_sentence = []
```

```
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
print(filtered_sentence)

lemma_word = []
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
for w in filtered_sentence:
    word1 = wordnet_lemmatizer.lemmatize(w, pos = "n")
    word2 = wordnet_lemmatizer.lemmatize(word1, pos = "v")
    word3 = wordnet_lemmatizer.lemmatize(word2, pos = ("a"))
    lemma_word.append(word3)
print(lemma_word)
```

view raw

nltklemma.py hosted with ❤ by GitHub

```
He determined drop litigation monastry, relinguish claims wood-cuting fishery rihgts. He ready becuase rights become much less valuable, indeed vaguest idea wood river question.
```

```
He determined drop litigation monastry, relinguish claim wood-cuting fishery rihgts. He ready becuase right become much le valuable, indeed vaguest idea wood river question.
```

> Here, **v** stands for **verb**, **a** stands for **adjective** and **n** stands for **noun**. The lemmatizer only lemmatizes those words which match the **pos** parameter of the lemmatize method.

Lemmatization is done on the basis of part-of-speech tagging (POS tagging). We'll talk in detail about POS tagging in an upcoming article.

2. Text Normalization using spaCy

spaCy, as we saw earlier, is an amazing NLP library. It provides many industry-level methods to perform lemmatization. Unfortunately, spaCy has no module for stemming. To perform lemmatization, check out the below code:

```
#make sure to download the english model with "python -m spacy download en"

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
```

view raw

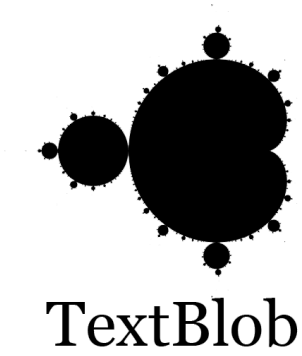
spacylemma.py hosted with ❤ by GitHub

```
-PRON- determine to drop -PRON- litigation with the monastry, and relinguish -PRON- claim to the wood-cuting 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.
```

Here *-PRON-* is the notation for pronoun which could easily be removed using regular expressions. **The benefit of spaCy is that we do not have to pass any *pos* parameter to perform lemmatization.**

3. Text Normalization using TextBlob

TextBlob is a Python library especially made for preprocessing text data. **It is based on the NLTK library**. We can use TextBlob to perform lemmatization. However, *there's no module for stemming in TextBlob*.



So let's see how to perform lemmatization using TextBlob in Python:

```
# from textblob lib import Word method
from textblob import Word

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."""

lem = []
for i in text.split():
    word1 = Word(i).lemmatize("n")
    word2 = Word(word1).lemmatize("v")
    word3 = Word(word2).lemmatize("a")
    lem.append(Word(word3).lemmatize())
print(lem)
```

view raw

textbloblemma.py hosted with ❤ by **GitHub**

```
He determine to drop his litigation with the monastry, and relinguish his claim to the wood-cuting and fishery rihgts at once. He wa the more ready to do this becuase the right have become much le valuable, and he have indeed the vague idea where the wood and river in question were.
```

Just like we saw above in the NLTK section, TextBlob also uses POS tagging to perform lemmatization. You can read more about how to use TextBlob in NLP here:

- Natural Language Processing for Beginners: Using TextBlob

End Notes

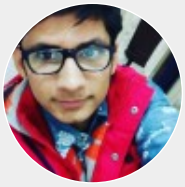
Stopwords play an important role in problems like sentiment analysis, question answering systems, etc. That's why removing stopwords can potentially affect our model's accuracy drastically.

This, as I mentioned, is part two of my series on 'How to Get Started with NLP'. You can check out part 1 on tokenization here.

And if you're looking for a place where you can finally begin your NLP journey, we have the perfect course for you:

- Natural Language Processing (NLP) Using Python

Article Url - https://www.analyticsvidhya.com/blog/2019/08/how-to-remove-stopwords-text-normalization-nltk-spacy-gensim-python/



Shubham Singh

A Data Science Enthusiast who loves reading & writing about Data Science and its applications. He has done many projects in this field and his recent work include concepts like Web Scraping, NLP etc. He is a Data Science Content Strategist Intern at Analytics Vidhya. And currently pursuing BTech in Computer Science from DIT University, Dehradun.